From sequences to knowledge, improving and learning from sequence alignments

Luc Blassel

SORBONNE UNIVERSITÉ

École doctorale Complexité du vivant, n. 515

Sequence Bioinformatics
Institut Pasteur/CNRS – USR3756

Thèse de doctorat en Génétique et génomique

From sequences to knowledge, improving and learning from sequence alignments

Luc Blassel

Dirigée par **Rayan Chikhi**

Présentée et soutenue publiquement le 2022-12-02

devant le jury composé de:

Brona BREJOVA	Associate Professor	Rapporteur
Macha NIKOLSKI	Group Leader	Rapporteur
Élodie LAINE	Associate Professor	Examinateur
Olivier GASCUEL	Directeur de recherche	Examinateur
Jean-Philippe VERT	Directeur de recherche	Examinateur
Paul MEDVEDEV	Associate Professor	Membre invité
Rayan CHIKHI	Group Leader	Directeur de Thèse

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

AA:	Amino Acid
AIDS:	Acquired Immunodeficience Syndrome
ART:	Antiretroviral therapy
BLAST:	Basic Local Alignment Search Tool
BLOSUM:	Block Substitution
CCS:	Circular Consensus Sequencing
CRF:	Circulating Recombinant Form
DNA:	Desoxy Ribonucleic Acid
dNTP:	deoxyNucleotide Triphosphate
DRM:	Drug Resistance Mutation
EI:	Entry Inhibitor
FI:	Fusion Inhibitor
HIV:	Human Immunodeficiency Virus
HMM:	Hidden Markov Model
HPC:	Homopolymer Compression
IN:	Integrase
INSTI:	Integrase Strand Transfer Inhibitor
LR:	Logisitic Regression
mapq:	mapping quality
ML:	Machine Learning
MSA:	Multiple Sequence Alignment
MSR:	Mapping-friendly sequence reduction
NNIBP:	Non-Nucleoside Reverse Transcriptase Inhibitor Binding Pocket
NNRTI:	Non-Nucleoside Reverse Transcriptase Inhibitor
NRTI:	Nucleoside Reverse Transcriptase Inhibitor
NTP:	Nucleotide Triphosphate
NW:	Needleman Wunsch algorithm
ONT:	Oxford Nanopore Technologies
PacBio:	Pacific Biosciences
PAM:	Point Accepted Mutation
PE:	Pharmacokinetic Enhancer
PI:	Protease Inhibitor
PR:	Protease
RAM:	Resistance Associated Mutation
RC:	reverse complement
RF:	Random Forest
RNA:	Ribonucleic Acid
RR:	Risk Ratio
RT:	Reverse Transcriptase
RTI:	Reverse Transcriptase Inhibitor
SSR:	Streaming sequence reduction
SW:	Smith Waterman algorithm

Acknowledgements

First, I would like to thank all the members of my jury for accepting to evaluate the work that I have done over the past few years. In particular, I would like to thank Macha Nikolski and Brona Brejova for reading my manuscript, reviewing it, and providing valuable feedback on my work. Second, I would like to thank the people that supervised me and helped me along the way. Thank you to Paul for your valuable insights. Thank you to Rayan for supervising me in these last years of this project, and to Olivier for doing so in the first. I would also like to thank Didier for helping me navigate the turmoil.

For most of my PhD, except for a few months in early 2020..., I spent my time on the 6th floor of the Omics building of the Institut Pasteur, in the *Evolutionary Bioinformatics* lab. Thank you to all my office mates for making my time on the 6th floor so pleasant. Thank you Anna, Fred and Jakub for all the discussions and coffee breaks. A special thanks to my good-***colleague*** Marie, who has ingrained in me the response of looking up and to the right whenever I hear an office chair move. Thank you also to the 6th floor adoptees: Jerome, Andrew, and Arthur, the earliest of them. I would also like to thank my labmates from the *Sequence Bioinformatics* lab: Camilla, Francesco, Yoann, Riccardo, Luca and Téo.

Of course, as studious as I might wish to be, I did spend 4 years at the office. I would therefore like to thank all my friends that have made them enjoyable. Grimault, Nathalie, Théo and Élise for the German escapades and the *(first of many I hope)* canyoning outings. Thank you as well to Eva M., Max, and Flore for being there throughout the years, and thank you to Eva C., Celia and all the other Spanish-sectionites for making me tolerate reggaeton. It has been a while since the Balzac days, but I am very glad we all still hang out! Dylan and Joshua, I started this adventure with you as roommates and enjoyed it very much, so thank you as well! Finally, thank you to Marie, Anna and Jean for many, many *(many!)* celebrations! Without all of you, my time as a PhD student would definitely have been less fun.

Next up on the thank you list is, of course, my family. I would like to thank my mom, dad and brothers, Noé and Tom, for supporting me along my journey. Thank you as well to my grandparents: Mamette, André and Grandma for believing in me and helping me along the way. To all the others from Tokyo to Santa Cruz, it warms my heart to know I have a worldwide support network, thank you!

Finally, Fatima. I am so glad I met you during this adventure. Thank you for supporting me, talking things through with me and helping me along the way. I

am very grateful that you are part of my life, and I feel better looking to the future knowing you will be by my side.

As parting words, I would like to dedicate this work to Pierre Blassel and John F. Murray. They made me, in part, what I am today and were essential in instilling in me the curiosity and scientific interest that are so necessary to pursue this type of endeavor. Although they could not see me finish, I would like to thank them from the bottom of my heart and I can only hope that they would be proud of me.

General Introduction

This manuscript is the result of my years at the Institut Pasteur, where I built upon work initiated during an internship in 2018. During my time at the Institut Pasteur I have worked on two very distinct subjects:

1. The study of drug resistance mutations in HIV sequences with machine learning.
2. The study of sequence transformation functions to improve long-read mapping.

These two subjects, though distinct, do share some common characteristics: mainly that they are based on sequence data and specifically alignments. Although the research on drug resistance in HIV was conducted before that on long-read mapping, I have forgone the chronological ordering of my work in this manuscript for the sake of thematic coherence. Through the organization of this manuscript, I have tried to link all the facets of my PhD work, and it is my hope that readers will be able to follow the flow without too much jumping around.

This manuscript is articulated around seven chapters, listed as follows:

1. An introduction to biological sequence data, how it is obtained and specific characteristics and problems inherent to long reads.
2. An introduction to sequence alignment, and how and why read-mapping is performed.
3. A presentation of my work on sequence transformation functions to improve long-read mapping, which was written as a standalone research article.
4. An introduction to machine learning on biological sequence data, with a focus on techniques used later in the manuscript.
5. An introduction to viruses and HIV in particular, with a focus on proteins important to drug resistance.
6. A presentation of my work on drug resistance in HIV, which was written and published as a standalone research article.
7. A short introduction to deep learning in sequence alignment and perspectives to the work presented in chapter 3.

Research output

During this thesis, my work on finding drug resistance mutations with machine learning resulted in two publications: a first author article describing our method published in *PLOS Computational Biology* as well as a co-first author review article published in *Current Opinion in Virology*.

The second half of my PhD work, on improving read-mapping resulted in a first-author paper, presented at the RECOMB-SEQ 2022 conference and to be published in the *iScience* proceedings of that conference.

In 2020, during the early stages of the COVID-19 pandemic and the lockdowns, I participated in some work resulting in the COVID-Align web-service and a middle-authorship in the corresponding *Bioinformatics* publication. This work also led to middle-authorship in an article concerning the origins of SARS-CoV-2 in the *Comptes Rendus. Biologies* journal of the French Science Academy.

Journal publications

This list contains the formal references of the publications mentioned above, along with my contribution represented using the CRediT taxonomy.

- **Blassel, Luc**, Paul Medvedev and Rayan Chikhi. 2022. **"Mapping-friendly sequence reductions: going beyond homopolymer compression"**. In press as part of the RECOMB-SEQ 2022 proceedings in *iScience*, *(Adapted as Chapter 3)*
  Contributions: Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.
- **Blassel, Luc**[1], Anna Zhukova[1], Christian J Villabona-Arenas, Katherine E Atkins, Stéphane Hué, and Olivier Gascuel. 2021. **"Drug Resistance Mutations in HIV: New Bioinformatics Approaches and Challenges."** *Current Opinion in Virology* 51 (December): 56–64. 10.1016/j.coviro.2021.09.009 *(Used as the basis for Section 5.3.4)*
  Contributions: Visualization, Writing – original draft, Writing – review & editing.
- **Blassel, Luc**, Anna Tostevin, Christian Julian Villabona-Arenas, MartinePeeters, Stéphane Hué, and Olivier Gascuel. 2021. **"Using Machine Learning and Big Data to Explore the Drug Resistance Landscape in HIV."** *PLOS Computational Biology* 17 (8): e1008873. 10.1371/journal.pcbi.1008873. *(Adapted as Chapter 6)*

[1]Co-first authors: Luc Blassel and Anna Zhukova

Contributions: Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

- Zhukova, Anna, **Luc Blassel**, Frédéric Lemoine, Marie Morel, JakubVoznica, and Olivier Gascuel. 2021. **"Origin, Evolution and Global Spread of SARS-CoV-2."** *Comptes Rendus. Biologies* 344 (1): 57–75. 10.5802/crbiol.29.
  Contributions: Writing – review & editing.

- Lemoine, Frédéric, **Luc Blassel**, Jakub Voznica, and Olivier Gascuel.2020. **"COVID-Align: accurate online alignment of hCoV-19 genomes using a profile HMM"** *Bioinformatics*, 37 (12): 1761-1762. 10.1093/bioinformatics/btaa871.
  Contributions: Software, Writing – review & editing

Presentations and posters

- **"Mapping-friendly sequence reductions: going beyond homopolymer compression"** proceedings talk, RECOMB-SEQ 2022. San Diego, USA *(May 21st 2022)*

- **"Can we improve analyses be transforming DNA?"** Joint RECOMB-SEQ RECOMB-CCB scientific communication session[2]. San Diego, USA (*May 21st 2022).*

- "**Machine learning approaches to reveal resistance mutations in HIV"** Poster at MCEB 2019. Porquerolles, France *(May 29th 2019)*

[2]2nd place prize awarded

1. What is Sequence Data ?

1.1. Biological sequences, a primer

To fully understand the work that was done during this thesis, as well as the choices that were made, some basic knowledge of molecular biology and genetics is needed. If you are already familiar with biological sequences, feel free to skip ahead to section 1.2.

1.1.1. What is DNA ?

Desoxyribo**N**ucleic **A**cid (DNA) is one of the most important molecules there is, without it complex life as we know it is impossible. It contains all the genetic information of a given organism, that is to say all the information necessary for the organism to: 1) function as a living being and 2) make a perfect copy of itself. This is the case for the overwhelming majority of living organisms on planet earth, from elephants to potatoes, to micro-organisms like bacteria.

DNA is a polymer, composed of monomeric units called nucleotides. Each nucleotide is composed of ribose (a five carbon sugar) on which are attached a phosphate group as well as one of four nucleobases: Adenine (A), Cytosine (C), Guanine (G) of Thymine (T). These four types of nucleotide monomers link up with one-another, through phosphate-sugar bonds, creating a single strand of DNA. The ordered sequence of these four types of nucleotides in strand encodes all the genetic information necessary for the organism to function. Nucleotides in a strand form strong complementary bonds with nucleotides from another strand, A with T and C with G. These bonds allow two strands of DNA to form the double-helix structure of DNA[1] shown in Figure 1.1. The specificity of nucleotide bonds ensure that the two strands of the double helix are complementary and that the information contained in one strand can be recovered from the other. This ensures a certain structural stability to the DNA molecule and a way to recover the important information that could be lost due to a damaged strand.

The amount of DNA necessary to encode the information varies greatly from organism to organism: 5400 base pairs (5.4kBp) for the $\varphi X174$ phage,[2] 4.9MBp for *Escherichia coli*,[3] 3.1GBp for *Homo sapiens*[4] all the way up to almost 150GBp for *Paris japonica,* a Japanese mountain flowering plant.[5] While very small genome size

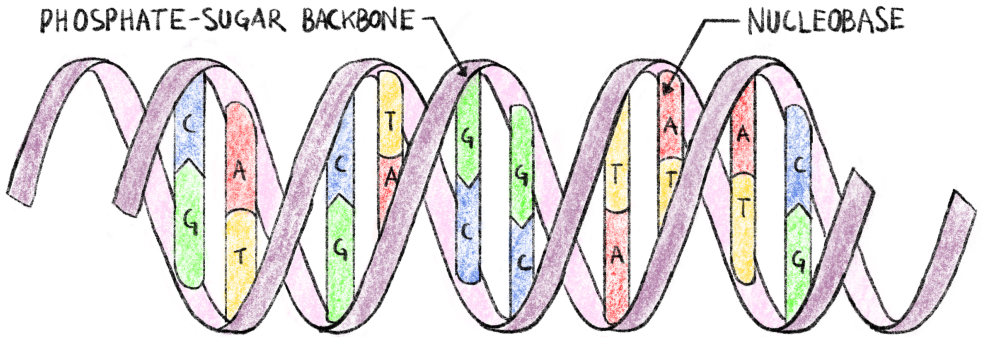


Figure 1.1.: **Double-helix structure of DNA.**
Each strand of DNA has a phosphate-sugar backbone on which are attached nucleobases. The two strands are linked by complementary bonds between the nucleobases of different strands (A bonding with T and C bonding with G), encoding the same information of both strands.

tend to occur in smaller, simpler organisms, genome size does not correlate with organism complexity.[6]

1.1.2. From Information to action

1.1.2.1. Proteins, their structure and their role

The double stranded DNA molecules present in the cells of a living organism contain information only; in order for the organism to live, this information must be read and translated into actions. Most of the actions necessary for "life" are taken by large molecules called proteins, they have a very wide range of functions from catalyzing reactions in the cell to giving it structure.[7]

Proteins are macromolecules that are made up of one or several chains of amino acids. These chains then link together and fold up in a specific three dimensional structure, giving the protein the shape it needs to fulfill its goal. This structure is determined by the sequence of amino acids, and a given protein can be identified by this amino acid sequence.[7]

This sequence is directly dependent on the information contained in the DNA. First the DNA is transcribed in a similar, but single stranded, molecule called RNA (Ribonucleic Acid) which encodes the same sequence. This RNA molecule is then translated into a protein by the following process:[8]

1. Nucleotides in the RNA sequence are read in groups of three called codons.

2. These codons are read sequentially along the RNA molecule.
3. Each codon corresponds to an amino acid, according to the genetic code.
4. The sequence of codons in RNA *(and by extension DNA)* determines the sequence of amino acids.
5. The translation process is stopped when a specific type of codon *(a "Stop" codon)* is read.

With four types of nucleotides and codons grouping three nucleotides there are $4^3 = 64$ possible codons. However, as stated above, proteins are only made up of 20 different amino acids, meaning that several different codons correspond to the same amino acid. This gives the translation process a certain robustness to errors that can occur when the DNA is copied to create a new cell, or when it is transformed into RNA prior to protein translation.

The portion of DNA that is read to create the protein is said to be "coding", and is called a gene. There are several thousands of genes in the human genome[9] resulting in proteins executing thousands of different functions in a cell. In human beings, coding DNA represents only 1% to 2% of the total genome.[10,11] The large majority of the DNA in a human being is not translated into proteins, a portion of it has a regulatory role, controlling transcription and translation, but the role remains unknown for the rest of the human genome.[12,13]

1.1.2.2. Making mistakes

Going from DNA sequence to protein is quite a complicated process involving several steps, it is therefore possible for a mistake to happen. There are several mechanisms to avoid mistakes and alteration of the genetic information: the complementary nature of the two strands of DNA, the redundant nature of the genetic code as well as error correction mechanisms in the molecules *(called "polymerases")* that read and write DNA and RNA being some of them. Despite all that, some errors in the nucleic acid (DNA and RNA) or protein sequences still make it through, these are called mutations.

1.1.2.2.1. Where can mistakes happen ? There are several sources of error that can alter genetic information:[14]

- **DNA replication:** When a cell divides, or when an organism reproduces, the DNA molecule must be copied in order to preserve and transmit genetic information. This process has a very low rate of errors, with as low as one error for every billion to every hundred billion of replicated base pairs.[15] This is due to the fact that the DNA polymerase (the protein responsible for copying DNA molecules), has a relatively low error rate to start with, but mostly to the error correcting mechanisms that are present in certain cells and bacteria.[16]

- **RNA transcription:** Since errors in RNA transcripts are less important than in replicated DNA, RNA polymerases have a much higher error rate than their DNA counterparts. This error rate has been estimated to be between four errors for each million[17] to two errors for each hundred thousand[18] transcribed bases.
- **Protein translation:** The process of translating RNA to a protein is done by proteins called ribosomes. This is a very error prone process with a mistranslation rate estimated to be of the order of one error for every 10,000 codons translated[19]
- **Other mutagenic events:** Many external events and factors have been shown to provoke mutations in exposed DNA such as Ionizing radiation,[20] UV rays,[21] toxins,[22] heat stress,[23] cold stress[24] or oxidative stress.[25]

While RNA transcription and protein translation are much more error prone processes than DNA replication, the errors induced only alter the expression of the genetic information. The effects of these errors are localized to the cells where they happen and are not transmitted to offspring. However these transcription errors are not unimportant and increased transcription error rates have been hypothesized to cause severe neurological symptoms in pediatric cohorts.[26]

1.1.2.2.2. What kind of errors are possible? In biological sequences (nucleic acids and proteins), mutations can result from one of three error modes:

- **Substitutions**, where the original base unit (nucleotide or amino acid) is mistakenly replaced by another one, for instance inserting an A instead of a G during RNA transcription.
- **Insertions**,where a new base unit not present in the original sequence is added to the newly synthesized biological sequence.
- **deletions**, where a base unit from the original sequence is skipped and not taken into account when synthesizing the new sequence.

While these three types of errors occur both in nucleic acids and proteins there are some things to consider about the consequences of nucleic acid mutations on protein synthesis. Due to the redundant nature of the genetic code mentioned in Section 1.1.2.1, some substitutions in the nucleic acid sequence will result in the same protein sequence and therefore not have altered protein activity. Some mutations however will result in a substitution at the amino acid level which could potentially lead to a physicochemically altered or even non-functional protein. Finally, insertion and deletion errors (collectively called indels) can have big consequences on resulting proteins. Inserting or deleting nucleotides in multiples of three will result in the insertion/deletion of amino acids in the protein, any other length of indel will result in what is called a frameshift mutation.[27] These mutations causes changes in all the codons following the mutation, potentially resulting in a completely different

amino-acid sequence, including premature stop codon apparition as shown in Figure 1.2.

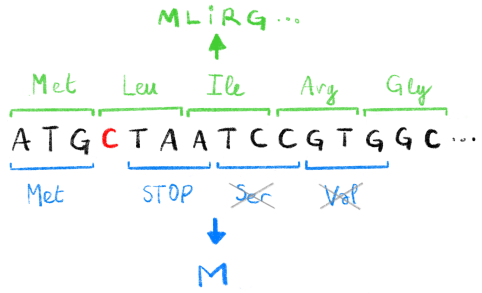


Figure 1.2.: **Effect of frameshift mutations.**
The deletion of a single C (highlighted in red) in the original DNA sequence leads to a change in the codons read during translation. The original codons (shown in green, with corresponding amino acids, above the sequence) translate to the functional protein `MLIRG...`. The new codons caused by the deletion (shown in blue, with corresponding amino acids, below the sequence), induce a premature STOP codon leading to a non-functional protein `M`. The Serine and Valine codons are not translated due to the STOP codon.

1.1.2.2.3. What effect can mutations have ? As we stated above, some mutations in DNA may have no repercussions, some others can lead to non-functional proteins. In some cases mutations can be associated with a trait in the mutated individual. For example a single mutation in a gene linked with coagulation can lead to pathological Leiden thrombophilia,[28] a single amino acid deletion in the CFTR protein leads to *(the very deadly)* cystic fibrosis,[29] and many mutations have been linked to complex diseases like type 2 diabetes.[30,31] All mutational effects are not necessarily bad for the organism though, and mutations are essential for bacteria[32] or viruses like HIV[33] to develop resistance to treatment *(more on that in Chapters 5 and 6).*

While some mutations, have had their mechanisms and consequences thoroughly studied, in many cases mutations are simply linked to a trait. Since it is easier to show correlation than causation, and that the former does not necessarily imply the latter, it is important to further study mutations of notice to understand their potential consequences.

1.2. Obtaining sequence data

In many fields, especially in computational biology, we need to know what genetic information the studied organism has. That is to say: what is the exact sequence

of nucleotides that make up its DNA? The process of figuring out this sequence is, perhaps unsurprisingly, called sequencing. A sequence that is produced by this process is called a *sequencing read* or, more commonly, just a *read*.

1.2.1. Sanger sequencing, a breakthrough

The first widely used sequencing method was developed in 1977.[34] Sanger *et al.* devised a simple method to read the sequence of nucleotides that make up a DNA sequence known as *chain termination sequencing* or simply *Sanger sequencing (represented in Figure 1.3)*. Although this method is now mostly obsolete, it established some key concepts in sequencing, some of which are in action in the most modern sequencers.

To understand Sanger sequencing, one must first understand how to synthesize DNA. As we stated in Section 1.1.1, DNA is built up from building blocks that we called nucleotides, more specifically deoxynucleotide triphosphates or dNTPs. These dNTPs are made up of a sugar (deoxyribose), a nucleobase (A, T, G or C) and 3 phosphate groups. By successively adding these dNTPs at the end of an existing DNA molecule, we extend it, linking one of the phospates of the dNTP to an oxygen atom on the last nucleotide of the DNA molecule. Let us now consider a dideoxynucleotide triphosphate (ddNTP), which is identical to a dNTP except we remove a specific oxygen atom. This ddNTP can be added to the growing molecule of DNA like regular dNTPs, but since it is missing that one oxygen atom no more dNTPs or ddNTPs can be added to the DNA molecule after this one. The elongation is terminated and we call these ddNTPs chain-terminators. This combination of DNA synthesis followed by termination are at the heart of Sanger sequencing.

It is important to note that dNTPs and ddNTPs refer to nucleotides with any nucleobase. We can refer to specific dNTPs by replacing the "N" with the base of choice. For example, dATP refers to the dNTP that has adenine as a base. Similarly we have dCTP, dGTP and dTTP (as well as ddATP, ddCTP, ddGTP and ddTTP).

1. The first step of Sanger sequencing (and most sequencing methods) is to amplify the DNA molecule we wish to sequence, *i.e.* make many copies of it (usually through a process called PCR). These clones of the sequence are then separated into their two complementary strands one of which will be used as a template for the sequencing steps.
2. The second step is to prepare 4 different sequencing environments *(think of it as 4 test tubes)*. In each environment we introduce an equal mix of the 4 dNTPs, that will be used to elongate new DNA molecules from the amplified templates, and a single type of ddNTP. So in the first test tube we will have only ddATP, ddCTP in the second, *et cetera.* In addition, these ddNTP are marked, at first with radioactive isotopes, and later on, as the technology

matured, with dyes. This marking means that we can observe the location of these ddNTPs later on.

3. Then an equal portion of the template is introduced in each environment with DNA polymerases (that will add the nucleotides to elongate a sequence that is complementary to the template), and short specific DNA molecules called primers that are necessary for the polymerases to start synthesizing new DNA.
4. During synthesis the chain is elongated with dNTPs by the polymerase and the reaction stops once a ddNTP is incorporated. At the end of this process we have plenty of fragments of DNA in each test tube, and we know that these fragments end with a specific base in a given environment. For example, in the test tube where we added ddATP, we know that all the fragments end with an A, and that we have all the possible fragments that start at the beginning of the template and end with an A. If the template is AACTA, then the fragments we would get in the ddATP test tube would be A, AA, and AACTA.
5. Then, a sample from each environment is taken and deposited on a gel, each in its own lane. A process called electrophoresis is then used to separate the fragments according to their weight. By applying an electrical current to the gel, the fragments of DNA will migrate away from where they were deposited along their lane in the gel. Lighter, shorter DNA fragments will travel further than heavier ones. We then get clusters of fragments ordered by weight (and therefore by length) called bands. With the marked ddNTP we can reveal these bands in the gel.
6. We know that: 1) bands are ordered by length; 2) consecutive bands correspond to the addition of a single nucleotide; 3) in a specific lane fragments corresponding to a band end with a specific base. This knowledge is enough to deduce the sequence of the template. An example gel is shown in Figure 1.3.

This process allowed Sanger *et al.* to sequence the first genome, of a $\varphi X174$ bacteriophage, in 1977.[2] Although revolutionary, this method was costly, time consuming and labor intensive. Adjustments to this method were made in order to make it faster and less expensive. An important step was to change the way ddNTPs were marked. By using fluorescent markers, each base having a distinct "color", we can eliminate the need to have 4 different environments and lanes in the gel.[35,36] This also paved the way for automating sequencing, each fluorescently marked band can be excited with a laser, and the resulting specific wavelength can be recorded by optical systems and the corresponding base automatically deduced[37] (Also see Figure 1.3). Other improvements were made such as using capillary electrophoresis instead of gel electrophoresis.

These gradual improvements to the Sanger sequencing protocol made it possible to sequence longer and more accurate reads, with the latest technologies resulting in reads reaching 1 ,000 base pairs with an accuracy of 99.999%.[38] These improvements also resulted in a lower cost for sequencing, which was greatly decreased from

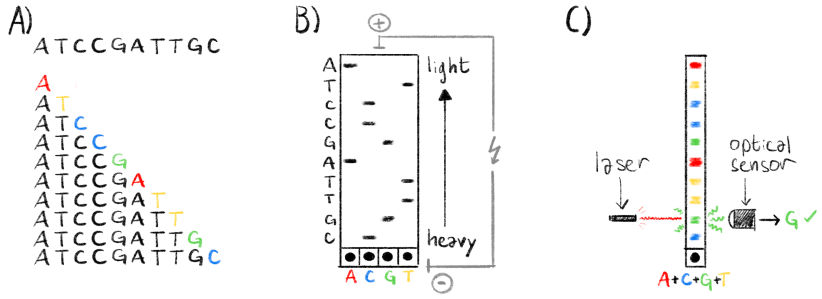


Figure 1.3.: **Overview of the sanger sequencing protocol.**
A) The sequence to read and all the generated fragments, with highlighted ddNTP chain terminators, ordered by molecular weight (i.e. length). **B)** Classical Sanger sequencing. The fragments are separated by electrophoresis and the lighter fragments travel further from the wells at the botom of the gel. Each lane in the gel correpsonds to a specific ddNTP. The radioactivly marked ddNTPs appear as black band in the gel and we can reconstruct the sequence by reading the bands from top to bottom, the column in which the band appears indicating which base is at each position. **C)** Automated Sanger sequencing. The fragments are also separated by electrophoresis, as in panel B. Chain terminators are marked with fluorescent markers. When excited by a laser, each ddNTP emits a specific wavelength. This is read by an optical sensor and the corresponding ddNTP is recorded. By exciting each band we can quickly deduce the sequence.

around $1000 per base-pair[39] to only $0.5 per kilobase.[38] Finally these technological improvements also increased the throughput of sequencing machines from around 1 kilobase per day[39] to 120 kilobases per hour.[40]

Despite these improvements, for ambitious endeavors such as the human genome project, sequencing was a massive undertaking: the first human genome is estimated to have cost between 500 million and 1 billion US dollars to sequence.[41]

1.2.2. Next-generation sequencing

Thanks to these large sequencing projects and the genomics field in general, the richness and usefulness of sequence data was made ever more apparent. This growing need of sequence data ushered in a new era of sequencing with the development of many new sequencing methods designed to have a higher throughput and a lower cost than Sanger sequencing. This second generation of sequencing technologies is often referred to as *Next-Generation Sequencing* (NGS) or *Massively parallel* sequencing. While there are different technologies, they share a few common key points:[42]

- As with Sanger sequencing, we first need to amplify and clone the DNA template. However, since these technologies result in shorter reads than Sanger sequencing, the DNA we want to sequence must first be randomly broken up into small template fragments before being amplified.
- The amplified template fragments are attached to some sort of solid support, resulting in a physical support with billions of template fragments attached to it.
- As in Sanger sequencing, DNA molecules, complementary to the template fragments, are elongated. This happens for billions of fragments at the same time (hence the "massively parallel" epithet).
- The addition of specific nucleotides to a chain are detected in real time, and there is no permanent chain termination. There is no need for the long step of electrophoresis. These detections are simultaneous for all the molecules being elongated at once.

The result of these steps is a very large number of short reads. With data analysis these short reads can be used to deduce longer sequences and eventually a fragmented approximation of the original whole genome sequence through a process called *assembly*.

The main NGS method is called "sequencing by synthesis", developed by a company: Illumina. It is commonly referred to as *Illumina sequencing*. This method is based on *reversible chain terminators*, developed at the Institut Pasteur in the 90's.[43] These are marked dNTPs that can be used to elongate DNA molecules, but that have an additional molecular group that makes them terminators by default. However this terminating group can be removed once the NTP is included in a DNA molecule allowing the elongation process to continue. These dNTPs are fluorescently marked and when excited with a laser they emit light with a distinctive color. During Illumina sequencing, these reversible chain terminators are included to millions of fragments at the same time, stopping elongation. At this point all the fragments are excited with a laser and an optical system takes a picture of the emitted colors for all the fragments at once. In this image, a pixel loosely corresponds to a sequenced fragment, and its color to the most recently added dNTP. The terminating groups are then cleaved and the process can start over by incorporating a new batch of reversible terminators. By observing the successive images we can deduce the sequence of added nucleotides for each sequenced fragment and obtain all of our reads.

Another NGS method is called pyrosequencing, commercialized by 454 Life Sciences. Contrary to Illumina sequencing, this method does not use reversible chain terminators. Instead it uses a special enzyme called luciferase that emits light as specific dNTPs are added. This process is repeated for the 4 dNTPs (similarly to Sanger sequencing) and from the light emissions we can deduce the sequence of nucleotides.[44]

These technologies yield reads around 150 nucleotides for Illumina and 400nt for pyrosequencing.[45] This is much shorter than the 1kB reads obtainable with the latest Sanger sequencing technologies. However the throughputs are much higher:[40] 2.5 to 12.5 Gigabases per hour for Illumina and 30 Megabases per hour for pyrosequencing. Costs are also quite low: \$0.07 and \$10 per Megabase for Illumina and pyrosequencing respectively. The per-base sequencing accuracies are also quite high, up to 99.9% for both Illumina[46] and pyrosequencing.[40] A summary of the key characteristics for various sequencing technologies can be found in Table 1.1. The lower cost and higher throughput has made the Illumina sequencing technology the dominant one. The company estimates that 90% of the world's sequencing data was generated with Illumina machines in 2015.[47]

1.2.3. Long read sequencing

Although NGS technologies revolutionized the sequencing world, recent efforts have been made to get longer reads. These third-generation methods generate reads of tens of kilobases and are commonly called *long-read sequencing* method. Long reads have a host of applications[48] for which short NGS reads might not be well suited: *De novo* assembly of large complex genomes, studying complex repetitive regions such as centromeres or telomeres or detection of structural variants. They have recently been used to assemble the first truly complete human genome, including telomeric and centromeric regions.[4]

The two available long read technologies are: Single Molecule Real Time sequencing (SMRT), commercialized by Pacific Biosciences (PacBio) and Nanopore sequencing, commercialized by Oxford Nanopore Technologies (ONT). While these technologies are quite different, they both result in much longer reads than even Sanger sequencing in real time, without the need for chain terminators or separate sequencing reactions, all with a high throughput and at a reasonably low cost.

SMRT sequencing was first developed in 2009,[49] before being commercialized and furthered by PacBio. The basic principle is as follows:

1. Fragment and amplify DNA to obtain a very large number of DNA templates.
2. Link both strands of each DNA template together with known sequences called *bell adapters*. Denature the DNA to create a single stranded, circular DNA molecule.
3. Primers and polymerases are attached to the circular molecule specifically on one of the bell adapters.
4. Add the circular DNA template, primer, polymerases complexes to a SMRT chip. This chip is essentially a large aluminium surface with hundreds of thousands of microscopic wells called *Zero-Mode Waveguides* (ZMWs) only 100nm in diameter.[50] The polymerases are chemically bonded to the bottom of each

of these ZMWs so we effectively get a single DNA template and polymerase per well.
5. Fluorescently marked dNTPs are incorporated progressively in each of the wells. When a marked dNTP is incorporated in the newly synthesized DNA brand, light of a specific wavelength is emitted.
6. The size of these ZMWs make the detection of the fluorescence possible with an optical system. Incorporation of dNTPs in each ZMW can be detected simultaneously in a parallel fashion and the resulting sequences deduced.

Nanopore sequencing, thought of in the eighties, further developed along the years[51] and first commercialized by ONT in 2014,[52] is completely different from all the sequencing technologies previously mentioned. Where all the other ones are based on synthesizing a complementary DNA strand and detecting specific dNTP incorporation in some way or another, there is no synthesis in nanopore sequencing. The principle relies on feeding a single strand of a DNA template through a small hole in a membrane, a *nanopore*, at a controlled speed. As the nucleotides go through the nanopore, an electric current is formed between both sides of the membrane. This current can be measured and is specific to the succession of 5 to 6 nucleotides inside the nanopore channel at any given time. By looking at the evolution of the electric current as the DNA strand goes through the nanopore, we can deduce the sequence of nucleotides through a process called *base calling*. Base calling is usually done with machine learning methods, mainly artificial neural networks.[53] In the flow cells used in ONT sequencers, there are hundreds of thousands of nanopores, spread out over a synthetic membrane, allowing for massively parallel sequencing as well. Theoretically, since this method is not based on synthesis, the upper limit for read length is only limited by the length of the template, and in practice ONT sequencing produces the longest reads.

Both technologies yield long reads, the median and highest read lengths being 10 kilobases and 60 kilobases respectively for PacBio sequencing.[54] For nanopore the median read lengths of 10 to 12 kilobases[55,56] are similar to PacBio, but in it can also yield ultra-long reads of 1 up to 2.3 megabases long.[57–59] The length of the reads and parallel nature of these two technologies allow these sequencers to have truly massive throughputs. PacBio sequencers can sequence between 2 and 11 gigabases per hour and ONT from 12.5 gigabases per hour, up to a staggering 260 gigabases per hour using the latest ONT PromethION machines.[56] The cost of sequencing with these machines, while higher than for Illumina sequencers, remains reasonably affordable at $0.32 and $0.13 per megabase for PacBio and ONT respectively.[60] These characteristics are summarized in Table 1.1 along with other sequencing technologies.

The length, throughput and sequencing cost of both these technologies paint a pretty picture, and indeed they have proved useful in many settings, but sequencing accuracy is a problem with these technologies. The per-base sequencing accuracy has been estimated to be between 85% and 92% for PacBio sequencers and 87% to 98%

technology	read length (nt)	throughput (nt/hour)	cost ($/Mb)	accuracy
Sanger	1 000	120 10^3	$500	99.999%
Illumina	150	2.5-12.5 10^9	$0.07	99.9%
Pyrosequencing	400	30 10^6	$10	99.9%
PacBio SMRT	10 000 (up to 60 000)	2-11 10^9	$0.32	85-92%
Nanopore	12 000 (up to 2.5 10^6)	12.5-260 10^9	$0.13	87-98%

Table 1.1.: **Comparison of sequencing technology characteristics.** Characteristics for the latest sequencers were used for the Sanger sequencing entry. The length is given in nucleotides, throughputs in sequenced nuctleotides per hour and cost in US dollars per megabase.

for ONT machines.[56,61,62] This accuracy is much lower than either Sanger sequencing or Illumina reads. Characterizing, correcting and accounting for these errors is widely studied and it will be discussed in more detail in Sections 1.3 and 1.4.

While most of the mentioned technologies can also be adapted and used to sequence RNA instead of DNA,[63,64] directly sequencing proteins remains a challenge. The sequence of amino acids making up a protein is usually deduced from the codons in sequenced DNA or RNA after detection of potentially coding regions called open reading frames (ORFs). Development of methods to directly sequence protein molecules using mass spectrometry was started not very long after Sanger sequencing[65] and improved.[66] New methods are still being developed[67] but protein sequencing remains a challenge.

1.3. Sequencing errors, how to account for them ?

Sequencing technologies are not perfect. They make mistakes, as we can see from the accuracy rates reported in Section 1.2. For technologies based on nucleic acid synthesis (i.e. everything except ONT), since they use polymerases it stands to reason that the same three types of errors, described in Section 1.1.2.2, occur: substitutions, insertions and deletions. For long read technologies though, most of the errors do not come from the polymerase, but from signal processing used to deduce the sequence. Since both technologies execute single molecule sequencing, the signal to noise ratio is low[68,69] making base calling more complicated.

This explains the discrepancy in error rates between short and long read sequencing technologies: the former getting as low as 10^{-4} or 10^{-5} after computational processing[70] where the latter are between 10% and 15%. This high error rate long reads is bothersome and many efforts have been made to lower this error rate, computationally and technologically.

1.3.1. Error correction methods

The long read error-correction literature and toolset is rich and active.[71–73] There are two main ways to correct errors: 1) hybrid methods where high-accuracy short reads are used, and 2) non-hybrid methods where only the long-reads are used.

In Non-hybrid methods,[71,74] by finding regions that overlap fairly well between reads and taking the consensus of the overlapped regions (i.e. the majority nucleotide at each position), some errors can be eliminated. In many analyses and sequencing data processing pipelines, the first step is to break up the reads into all possible overlapping subsequences of length k called k-mers (e.g the 3-mers of ATTGC are ATT, TTG and TGC). Rare k-mers in the read dataset, i.e. k-mers that appear only a handful of times in all the reads, are likely the result of an error and filtering them out can improve analysis. One or both of these procedures are implemented in several pieces of commonly used software such as assembler like `wtdbg2`,[75] and `canu`[76] or standalone long-read correctors like `daccord`.[77] In some cases, errors are corrected not on the raw reads but after having assembled the long reads into long continuous sequences (contigs), this process is called polishing. The `ntEdit` polisher[78] also filters out rare k-mers to correct errors. The `Arrow`[79] and `Nanopolish`[80] polishers correct the assembly using the raw PacBio and ONT long reads respectively, and `Racon`[81] can use bot types of long-reads to polish assemblies.

Hybrid methods, as their name suggest, make use of short reads to correct errors in long reads. By finding similar regions between the short and long reads we can use the higher accuracy of short reads to correct the long ones. This is implemented in many pieces of software `proovread`,[82] `Jabba`,[83] `PBcR`[84] or `LoRDEC`.[85] Short reads can also be used to polish long read assemblies with tools like `Pilon`.[86] The first complete human genome was assembled and polished using many different sequencing technologies including PacBio, ONT and Illumina technologies.[4]

1.3.2. More accurate sequencing methods

While a lot of effort is being put into error correction, another angle of attack to lower the error rate of long reads is to improve the sequencing technology.

In 2019, PacBio introduced HiFi reads, based on a circular consensus (CCS) technique.[87] During SRMT sequencing the 2 strands are linked together by bell adapters to form a circular DNA template (c.f. Section 1.2.3), the central idea of CCS is to sequence this molecule multiple times by going over the circle more than once. In the resulting long sequence the known bell adapter sequences can be removed, and a consensus sequence can be built from the multiple passes over the same DNA template. This results in long-read accuracies of 99.8% to 99.9%.[56,87] This works because

PacBio sequencing errors are mostly randomly distributed along the sequenced template (more on that in Section 1.4.2). Therefore it is unlikely that the same error will appear in multiple passes over the same template portion.

For ONT sequencing, most improvement efforts have been focused on base-callers. These tools were originally based on Hidden Markov Models[88] (HMMs), but gradually they have been shifting over to neural network based deep learning methods[53,74,89,90] with faster inference times and better performance.

Similarly to PacBio HiFi reads, ONT developed 2D, and $1D^2$ sequencing. In 2D sequencing, both strands of the DNA molecule to sequence are linked with a hairpin adapter to form one long sequence. Each strand is sequenced once and a consensus is built from these 2 passes.[91] $1D^2$ sequencing operates in a similar fashion but without the need for a hairpin adapter.[92] 2D sequencing produces reads with 97% accuracy albeit much shorter than standard 1D sequencing.[91] Recently, Oxford Nanopore Technologies announced the release of a new technology they call duplex. Using new chemistry, a new basecaller and sequencing of both strands (similarly to 2D and $1D^2$) they announce raw read accuracies of 99.3%.[93] Pre-printed research seems to confirm these numbers with one experiment yielding duplex reads with a 99.9% accuracy.[94]

A technologically agnostic method using unique molecular identifiers added during the template preparation phase, and consensus sequencing has been shown, in specific contexts, to improve the accuracies of both ONT and PacBio CCS long reads to 99.59% and 99.93% respectively.[95]

Finally, new sequencing technologies are being developed, like built in error-correction short-read technologies yielding error-free reads of up to 200 nucleotides long.[96] Illumina also recently announced its own high-throughput, high-accuracy long-read sequencing technology in 2022,[97] although details about the performance and technology are scarce.

1.4. The special case of homopolymers

Despite improvement in error correction methods and sequencing technologies, certain genetic patterns are particularly difficult to process, homopolymers are one such pattern.

1.4.1. Homopolymers and the human genome

Homopolymers consist of a stretch of repeated nucleotides (i.e. ≥ 2) occurring at some point in the genome. For example, the sequence AAAA is a length 4 adenine homopolymer. In the complete human genome assembly (CHM13 v1.1 from the T2T consortium[4]), 50% of its three gigabases are in homopolymers of size 2 or more,

and 10% are in homopolymers of length equal or greater than 4. As can be seen in Figure 1.4, short and medium length homopolymers make up a significant part of the genome. In a previous GRCh38 human genome assembly, more than 1.9 megabases are in homopolymers of length 8 or higher,[98] representing about 1‰ of that assembly. The longest homopolymer run in the CHM13 v1.1 assembly is 86 *(90 in GRCh38[98])*.

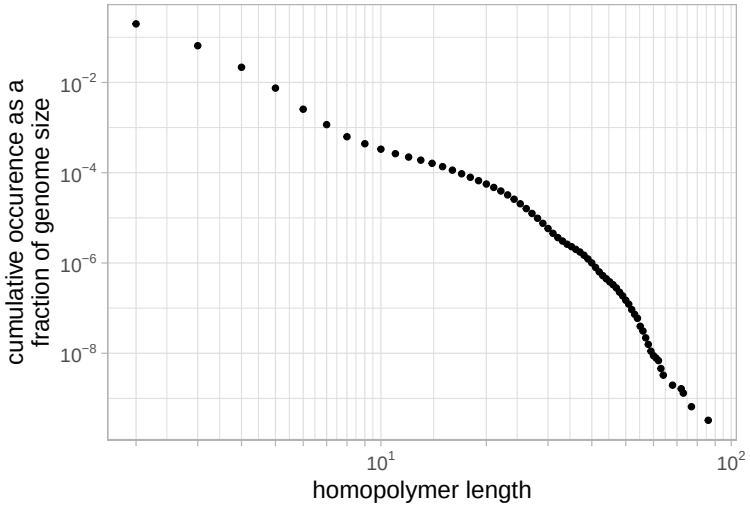


Figure 1.4.: **Homopolymer fraction of the whole human genome by homopolymer length.**
The homopolymer counts were calculated from the T2T consortium full human genome assembly CHM13 v1.1. This figure was inspired by Figure 3b of reference.[98]

In the human genome, homopolymers tend to occur more often in adenine and thymine runs than guanine and cytosine. There are are approximately twice as many nucleotides within A or T homopolymers (481 Mb and 484 Mb) than G or C (278 Mb and 279 Mb). This discrepancy is even more pronounced when looking at homopolymers longer than four nucleotides (c.f. Figure 1.5).

1.4.2. Homopolymers and long reads

Unfortunately, homopolymers are a source of errors in sequencing, particularly for long-read technologies. While substitutions seem to be randomly distributed along the reads for PacBio and ONT, the main error mode seems to be indels in homopolymeric sections, *i.e.* reading the same nucleotide several times or skipping over one of the repeated nucleotides. Many studies show that homopolymeric indels are the main type of error for PacBIO SMRT and ONT long-read sequencing.[68,99–101] This is even the case for PacBio HiFi reads, while the circular consensus approach eliminates the randomly distributed substitutions but homopolymer indels remain.[87] It seems that ONT reads are more prone to this type of error than PacBIo.[56] The rate

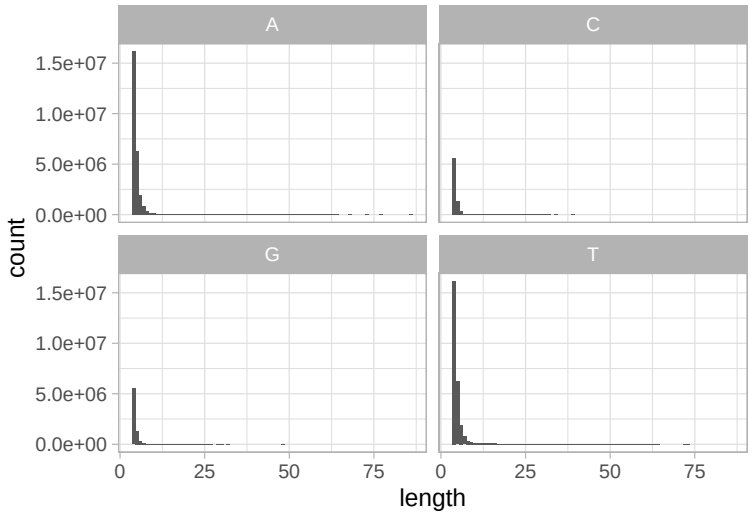


Figure 1.5.: **Distribution of homopolymer lengths per base in the human genome, for homopolymers of length ≥ 4.**
The homopolymer counts were calculated from the T2T consortium full human genome assembly CHM13 v1.1.

of these errors is independent of the length of the homopolymer for ONT, but it rises with homopolymer length for short-read and PacBio technologies.[102]

1.4.3. Accounting for homopolymers

The fact that they make up a significant part of the human genome, and that they are a source of errors for long read technologies means that homopolymers warrant special attention and care. Methods have been devised and implemented, specifically to counter homopolymer-linked errors.

1.4.3.1. Specific error correction

Homopolymer errors are taken under special consideration during assembly polishing when using certain tools like `HomoPolish`,[103] `NanoPolish`[80] or `Pilon`.[86] Methods to improve base calling of homopolymer stretches have been developed for nanopore sequencing,[104,105] and implemented in state of the art base-callers such as `guppy` or `scrappie`.[53]

Steps before sequencing can also be taken in order to reduce the effect of these errors, like avoiding homopolymers in barcode sequences[106,107] or during the development of DNA based storage systems.[108]

Improving the sequencing technologies can also be a solution, by reducing the number of homopolymer errors straight from the source. The latest ONT chemistry R.10 reportedly improves accuracy in homopolymer rich regions.[74,109] Non-biological solid-state nanopores also reduces errors in homopolymers.[110,111]

1.4.3.2. Homopolymer compression, a nifty trick

Homopolymer-errors can be harmful for downstream analyses such as read-mapping (c.f. Chapter 3). However, in many cases, reads cannot be re-sequenced with newer technologies, or base-called with better base callers. Only the read sequences potentially containing homopolymer errors, are available for usage. In order to account for this sort of error, a simple pre-processing trick was developed: *homopolymer compression* (HPC).

The idea is very simple: for any sequence, replace a repeated run of any nucleotide (i.e. homopolymers) by a single occurrence of that nucleotide. This means that after going through HPC the sequence AAACTGGG will yield the sequence ACTG. This simple pre-processing step, applied to all the reads and sequences to analyze, removes all indels in homopolymers, and can resolve some ambiguities (c.f. Figure 1.6). It can also remove legitimate information contained in homopolymers, but the trade-off with the reduced error rate has been deemed advantageous.

HPC has been implemented in many sequence bioinformatics software tools. The `HiCanu`,[112] `MDBG`,[113] `wtdbg2`,[75] `shasta`[114] assemblers all use HPC under the hood to provide better assemblies, and HPC was used to assemble the complete human genome sequence.[4] The first published usage of HPC, was actually in the `CABOG` assembler[115] developed for pyrosequencing reads. HPC has also been implemented for other tasks, like clustering,[116] long read error correction with `LSC`[117] and `LSCPlus`,[118] alignment with `minimap2`[119] and `winnowmap2`,[120] or specific analysis pipelines for satellite tandem repeats.[121]

1.5. Conclusion

I hope, after reading this chapter, you will agree with me that sequencing is fundamental for furthering our knowledge of biological processes, organisms and Life in general. And as such, the sequencing field is still very active with new technologies being developed to improve the current technologies in various aspects. Illumina promises high accuracy long reads with Infinity[97] and PacBio is developing its own short read sequencing technology, moving away from sequencing by synthesis.[122,123]

[a] Homopolymer indels can be harmful in opposite circumstances as well. Let us consider, for example, a read that should correspond to several repetitions of a conserved motif. Homopolymer indels can artificially resolve an ambiguity by making the read unique and prefer a specific repetition of the motif or entirely misplace the read.

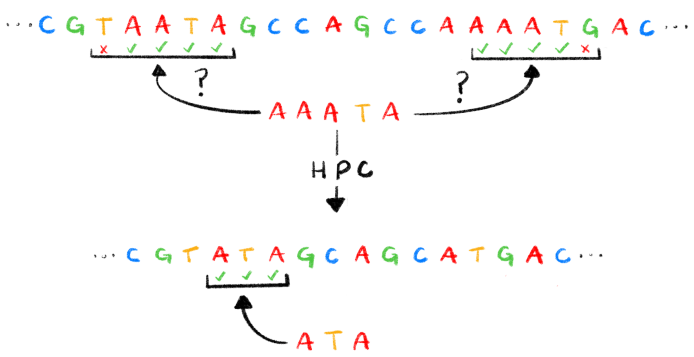


Figure 1.6.: **Homopolymer compression can help resolve ambiguities due to sequencing errors.**
A read with homopolymer related sequencing errors can be homologous to two different regions of the reference genome, with one discrepancy for each region. After applying HPC, this ambiguity is properly accounted for and the read is homologous to only one region. This figure, however, only shows one way homopolymers can be detrimental and others are possible[a].

Finally, efforts are also being made to make sequencing more affordable and available in a greater number settings with Ultima genomics promising accurate short reads for as low as $1 per gigabase.[124]

With all these technological improvements we are approaching an era where sequencing is easy and quick, opening the door for massive projects like Tara Oceans[125] or the BioGenome project[126] to better understand biodiversity. Routine whole-genome sequencing could also usher in an era personalized medicine.[127]

Despite all these advancements, sequencing errors remain an obstacle to certain analyses. This is particularly true for the ever more used and useful long reads, and the important fraction of genomes made up of homopolymers. Detecting, removing or accounting for these errors in some way is a crucial step to improve any analysis based on sequencing data, and to make sure that no theory or conclusion are built upon erroneous sequence data.

Finally, it is important to note (at least for the remainder of this thesis) that, from a computational standpoint, a biological sequence is simply a succession of letters and a set of reads is simply a text file. Therefore, many analyses and data processing methods are inspired or directly transposed from the field of text algorithmics.

2. Aligning Sequence Data

2.1. What is an alignment ?

In biology, comparison is at the heart of many studies: between individuals, between species, between sequencing runs, *etc*... In order to do this at a fine-grained level and extract knowledge from it, we need to compare what is comparable, this is where sequence alignment steps in. In broad terms, during sequence alignment, we aim to find regions similar to each other in two or more sequences and group them together. When this process is done with only two sequences it is called a *pairwise alignment*. When three or more sequences are used it is called *multiple alignment*. We will first focus on pairwise alignment as it is used as the basis for the more complex multiple alignment.

2.1.1. Why align ?

The first question we might ask ourselves is why align at all? If we want to compare two sequences there are plenty of distances and metrics out there to use. Something like the Hamming distance[128] is very quick and easy to compute by comparing characters two by two. It is however ill-suited to our needs in biology: it can handle substitutions but indels induce very large Hamming distances. Indeed, insertions and/or deletions shift one of the sequences, compared to the other, and introduce many character-to-character differences that could be explained by a single indel.

For example, let us consider the two following sequences: `ATGTGCAGTA` and `AGTGCAGTAC`. if we count the differences character by character, except the first pair of A, all the characters are different (c.f. below). However, if we consider that the first T was deleted and a C was inserted at the end of the second sequence then we can see that none of the characters are actually different. In order to represent insertions and deletions *gaps* are inserted in the sequences as seen below:

```
ATGTGCAGTA-
A-GTGCAGTAC
```

This problem of comparing two sequences with insertions or deletions is a fairly well-studied one in text algorithmics: the string-edit problem.[129] Some metrics like the Levenshtein distance[130] and the edit distance[129] exist and are closely related to the

pairwise sequence alignment problem, finding the minimal number of substitution, insertion of deletion operations to go from one sequence to the other.

Sequence alignments have many downstream uses. They are the basis of comparative genomics[131] and are used to infer evolutionary relationships, phylogenetic tree reconstruction methods usually take multiple sequence alignments as input.[132–136] Sequence alignments have been used to study protein structure[137,138] and function.[139,140] They can be used to correct sequencing errors[82,84,141] or detect structural variations in genomes.[142,143] All this to say that they are absolutely fundamental to the field of computational biology and errors in alignments can lead to errors somewhere down the line.

2.1.2. How to align two sequences ?

There are two approaches for pairwise alignment:[144] *global alignment,* where the entirety of both sequences is used, and *local alignment,* where we only seek to find regions in each sequence that are most similar to each other. Global alignment is used when the two sequences are expected to be quite similar (e.g. comparing two related proteins), whereas local alignment is mostly used when we expect the sequences to be fairly different but with highly similar regions, like genomes of two distantly related species that share a highly conserved region.

The seminal method for global pairwise alignment was the Needleman-Wünsch algorithm[145] based on a dynamic programming method. A decade later, the Smith-Waterman algorithm[146] was developed with similar ideas to perform local alignment. Both are still used today for pairwise alignment.

Dynamic programming is often used to solve complex problems by breaking it into smaller sub-problems and solving each one optimally and separately,[147,148] it is particularly useful when we wish to have a precise alignment between 2 sequences.

2.1.2.1. Global alignment

The fundamental algorithm for globally aligning two sequences is the Needleman-Wünsch (NW) algorithm.[145] The goal is finding the alignment with 1) the lowest edit-distance or 2) the highest alignment score. These two are equivalent so in this section we will maximize the alignment score.

The first thing we need to know is how to compute a score on a given alignment. To do this, we assign costs to each operation. Usually matches (i.e. aligning two identical characters) are given a positive cost and mismatches or indels a negative cost. If we assign a cost of +1 to a match and a cost of -1 to mismatches and indels then the alignment presented above in Section 2.1.1 would have an alignment score of 9 - 2 = 7 *(9 matches and two indels).*

The NW algorithm is based on a simple recurrence relation: the optimal alignment score of two sequences S_1 and S_2 of lengths n and m respectively is the maximum of:

1. The optimal alignment score of $S_1[1, n-1]$[a] and $S_2[1, m-1]$ plus the cost of a match or mismatch between the n^{th} character of S_1 and the m^{th} character of S_2
2. The optimal alignment score of S_1 and $S_2[1, m-1]$ plus the cost of an indel
3. The optimal alignment score of $S_1[1, n-1]$ and S_2 plus the cost of an indel

This simple relation can be used to compute optimal global alignment score for two sequences. However, if it is implemented naively it can be very inefficient as the number of scores to compute grows exponentially with sequence lengths, and many intermediary alignment scores need to be computed many times. This is where dynamic programming comes in: these intermediary costs are pre-computed in an efficient manner and one can then deduce the optimal alignment from these. This pre-computing step is usually represented as filling out a matrix, whose rows and columns represent the characters in each sequence to be aligned, with partial alignment scores.

If S_1 represents the rows of the matrix, and S_2 the columns, the value $C(i, j)$ of a cell (i, j) of this matrix represents the optimal alignment score between $S_1[1, i]$ and $S_2[1, j]$. In the recurrence relation described above the alignment score as dependent on the optimal alignment scores of subsequences, when filling out the dynamic programming matrix we proceed in the inverse fashion by using the scores of short subsequences to build up the scores of progressively longer sequences.

We will go here through a short example showing how the NW algorithm is used to align two short sequences: S_1 =`ACCTGA` and S_2 =`ACGGA`. The first step is to represent the dynamic programming matrix, prefix each sequence with an empty character and label the rows of the matrix with one of the sequences and the columns with the other *(this extra row and column at the beginning of each sequence are indexed as column and row 0)*. In this matrix, due to the recurrence relation stated above, the score of a particular cell, $C(i, j)$, is the maximum of:

1. The score in the diagonally adjacent cell $C(i-1, j-1)$ plus the cost of a match or mismatch between $S_1[i]$ and $S_2[j]$.
2. The score of the cell to the left $C(i, j-1)$ plus the cost of an indel
3. The score of the cell on top $C(i-1, j)$ plus the cost of an indel

Therefore, in order to compute $C(i, j)$ we need to know the three values of $C(i-1, j-1)$, $C(i-1, j)$ and $C(i, j-1)$. This is the reason why we start with an extra column

[a] Here I am using an index starting at 1 and inclusive, so $S_1[1, n-1]$ represents the first $n-1$ characters. If $S_1 = ABCD$ then $S_1[1; 3] = ABC$

and row at the beginning of each sequence that we can fill out with the increasing costs of indels. In our case since the cost of an indel is -1, this row and column are filled out with decreasing relative integers, as can be seen in Figure 2.1A.

From this starting point we can fill out the dynamic programming matrix with all the alignment scores. To compute $C(1,1)$ we have three possible values:

1. $C(0,0)$ plus the cost of a match between $S_1[1] = A$ and $S_2[1] = A$: $0 + 1 = 1$
2. $C(0,1)$ plus the cost of an indel: $-1 - 1 = -2$
3. $C(0,1)$ plus the cost of an indel: $-1 - 1 = -2$

By taking the maximum out of these three values we can fill out the matrix cell with $C(1,1) = 1$. By continuing this process until we fill out the whole matrix we obtain the scores visible in Figure 2.1A. This is enough if we only want to compute the optimal global alignment score between S_1 and S_2, contained in cell (n, m). If we want to deduce the operations leading to it, and therefore the alignment itself, we need to keep track of which operation we made to get a specific score. The easiest way to do that is to also consider this matrix as a graph where each cell is a vertex. When we compute the score of cell (i, j) we add an edge from this cell to the previous cell that was used to compute $C(i, j)$. In our example above, we obtained $C(1,1)$ from a match and $C(0,0)$, so we can add an edge in our graph going from cell $(1,1)$ to cell $(0,0)$. The filled out matrix with the graph edges represented as arrows can be seen in Figure 2.1B.

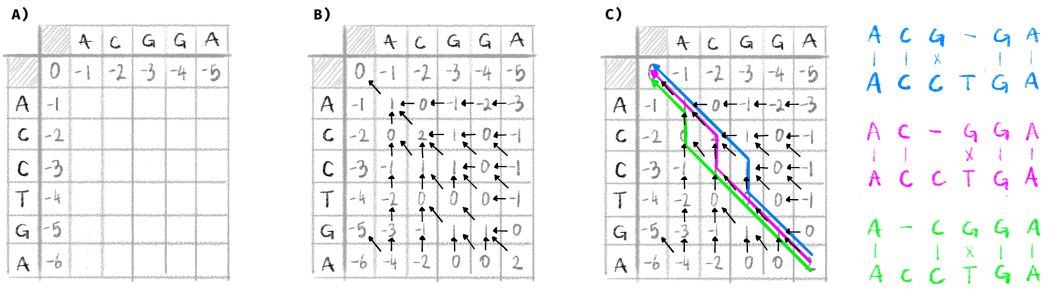


Figure 2.1.: **Example global alignment with the Needleman-Wunsch algorithm.**
This figure represents three different steps in the NW algorithm, with a match cost of +1, a mismatch cost of -1 and an indel cost of -1. **A)** the matrix is initialized with S_1 as the columns and S_2 as the rows. Column and row 0 are filled out. **B)** The dynamic programming matrix is filled out, and the alignment graph is constructed. **C)** The alignment graph is traversed from the vertex in the bottom right cell to the vertex in the top left cell. Each of the three possible paths corresponds to an optimal global alignment, represented on the right.

Once this matrix *(and corresponding graph)* is filled out, we can deduce the alignment by following a path through the graph starting at cell (n, m) to cell $(0, 0)$. A diagonal edge starting at (i, j) indicates a match or mismatch between $S_1[i]$ and $S_2[j]$, a vertical edge indicates a gap in S_2 and a horizontal edge a gap in S_1. This can lead to several optimal alignments if there are several such paths in the graph. In our case, this algorithm yields three equally optimal global alignments shown in Figure 2.1C.

This algorithm although guaranteed to result in an optimal alignment, has a time complexity of $\mathcal{O}(nm)$ where n and m are the lengths of the sequences to align.[144] Some methods have been proposed to speed up,[149] however the complexity is still $\mathcal{O}(nm/\log(n))$. Lower bounds have been studied and there is not much optimization to be done if optimal exact alignment is needed.[150,151] If we want to do better we have to rely on heuristics.

Another issue is space complexity since we need to store the matrix, the space complexity is also $\mathcal{O}(nm)$. If we wish to align 2 human genomes we would need to store $\approx 10^{19}$ matrix cells, which would amount to 10 Exabytes of storage *(i.e. the storage scale of a data-center)* if we use 8bit integers. However, in practice, we can do much better than that, and construct an optimal alignment in linear space complexity $\mathcal{O}(n + m)$[152] meaning we would only need a couple gigabytes to store the matrix for 2 human genomes. This resulted in an improved global alignment algorithm, the Myers-Miller algorithm,[153] implemented in the EMBOSS `stretcher` alignment software.[154]

2.1.2.2. Local alignment

In global alignment two full sequences are aligned to each other. In local alignment the goal is to find the optimal alignment of two subsequences from these parent sequences. The main algorithm for locally aligning is the Smith-Waterman (SW) algorithm,[146] developed a decade later than NW.

The two algorithms are very similar, SW also relies on first building the dynamic programming matrix with the same parametrizable costs for matches, mismatches and indels as NW. One key difference is that the optimal scores in the matrix are bound by 0 so they cannot become negative. We only store edges in the alignment graph if the starting cell has an alignment score > 0.

In this new formulation, the score in cell $C(i, j)$ is the maximum of the following values:

1. The score in the diagonally adjacent cell $C(i-1, j-1)$ plus the cost of a match or mismatch between $S_1[i]$ and $S_2[j]$.
2. The score of the cell to the left $C(i, j-1)$ plus the cost of an indel.
3. The score of the cell on top $C(i-1, j)$ plus the cost of an indel.

4. 0.

If we use the SW algorithm to locally align the two example sequences S_1 and S_2 and the same costs as used above, we obtain the dynamic programming matrix and graph shown in Figure 2.2.

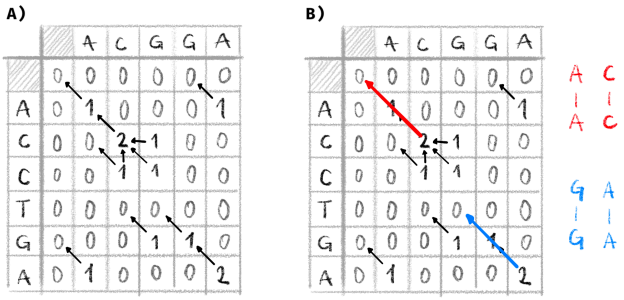


Figure 2.2.: **Example local alignment with the Smith-Waterman algorithm.** Two sequences S_1 and S_2 (the same as in Figure 2.1) are locally aligned. A match has a cost of +1, a mismatch a cost of -1 and indels a cost of -1. **A)** The dynamic programming matrix is filled out and the alignment graph constructed. Alignment scores are constrained to be non-negative. **B)** We find paths in the graph between the vertex with the maximal score and one with a score of 0. Here there are two such paths resulting in two optimal local alignments represented on the right.

The traceback part to determine the optimal alignment is very similar to NW, however instead of starting at cell (n, m), we start at the cell in with the maximal alignment score and follow the path back until we arrive at a cell with an alignment score of 0. In the example shown in Figure 2.2, two cells contain the maximal alignment score of 2. Tracing back from these cells gives two optimal local alignments between S_1 and S_2: `AC` to `AC` and `GA` to `GA`.

Since the SW algorithm is so similar to NW it has the same quadratic time and space complexity. However, the same optimization can be used to bring it down to a linear space complexity.[144] These optimizations resulted in the Huang and Miller algorithm,[155] implemented in the EMBOSS `Lalign` tool,[154] and the Waterman Eggert algorithm.[156]

Both the NW and the SW algorithms are implemented in many different software tools and are used widely to perform pairwise alignments of short sequences.[154,157,158] Some versions even benefit from hardware acceleration with version implemented for specific CPU instruction sets[159] or GPUs[160] to substantially speed up alignment.

2.1.3. Scoring and substitution models

In the examples used above to present the NW and SW algorithms, we used a very simple cost function: a match has a cost of +1 while mismatches and indels have a cost of -1. This is really the simplest cost function we can use but also the crudest. In many cases it may be interesting to infuse this cost function with biological knowledge. For example some substitutions occur more rarely than others in nature so it would stand to reason to penalize those more than other, more common, substitutions.

These biology-aware cost functions usually take the form of a matrix, called *scoring* or *substitution matrix*, often corresponding to an underlying evolutionary model. When using these matrices, matches and mismatches between specific characters are given. For example the cost of aligning an A and a G might be lower than aligning that same A with a T. A lot of different substitution matrices have been developed especially for protein alignments,[161] developed with different techniques and underlying models and with different intended applications.

The earliest and simplest substitution matrices are match/mismatch matrices. They are effectively what we used above, where all matches are given a fixed positive score and all mismatches a fixed negative score. In our examples above the corresponding substitution matrix would be a four by four matrix with ones on the diagonal indicating matches and -1 everywhere else. These are simple and useful, but when dealing with proteins, they have a severe limitation as they ignore the biology of amino acids.

In order to reflect this biological reality of proteins, new substitution matrices were developed using Log-odds models based on the fact that substitutions in amino acids are not equiprobable, and some mutations between related amino acids *(e.g. I and L)* are much more common than others. Two of the most widely used substitution matrices, PAM and BLOSUM matrices, were built this way. The score for aligning residue i with residue j is given by the matrix entry S_{ij} by looking at the background frequencies *(i.e. how often one expects to see a particular residue in a sequence)* of i and j denoted p_i and p_j respectively and the frequency q_{ij} with which i and j are aligned in accurate biological alignments. With these values we can compute the substitution score s_{ij} as a Log-odds:[161]

$$S_{i,j} = \log\left(\frac{q_{ij}}{p_i p_j}\right)$$

This Log-odds formulation yields values with nice properties for sequence alignment. q_{ij} can be thought of as the probability of the alignment between amino acids i and j resulting from a substitution, and $p_i p_j$ is the probability under the null hypothesis that both of these amino acids were aligned randomly. Therefore the log of the ratio is negative when the random alignment is more frequent (meaning the substitution

is unlikely), and positive when the substitution is likely. Both p_i and p_j are easy to compute from available biological sequence data, the real work in developing a Log-odds based substitution matrix is to estimate q_{ij} values, and that is often done using biologically accurate protein sequence alignments.

The PAM matrix, developed in 1978,[162] is one such matrix. A *point accepted mutation* (PAM) is defined as the substitution of one amino acid by another that is accepted by natural selection *(i.e. visible along the branch of a phylogenetic tree)*. Dayhoff *et al.* also defined a PAM as an evolutionary distance, where two sequences distant by one PAM are expected to have one amino acid substitution per one hundred residues, which is equivalent to expecting a substitution at 1% of positions. To develop their matrix, Dayhoff *et al.* used phylogenetic trees built on 71 families of closely related proteins and counted the PAMs that appeared in these trees. This resulted in a matrix A where $A_{ij} = A_{ji} =$ the number of times a substitution between amino acids i and j was observed in the trees. By using trees built on closely related sequences, Dayhoff *et al.* could be fairly certain that the observed substitutions were the result of a single mutation and not many subsequent mutations over long evolutionary times. From this matrix A, Dayhoff *et al.* reconstructed the mutation probability M_1 where entries $M_{1,ij}$ represent the probability of amino acid j being replaced by amino acid i after an interval of 1 PAM. Entries of this matrix are computed as follows:

$$M_{1,ij} = \frac{\lambda m_j A_{ij}}{\sum_i A_{ij}} \qquad \text{if } i \neq j \tag{2.1}$$

$$M_{1,ij} = 1 - \lambda m_j \qquad \text{if } i = j \tag{2.2}$$

here m_j is the observed mutability of amino acid j, and λ is a constant factor used to tune the matrix so that it reflects mutation rates corresponding to 1 PAM where 99% of positions are unchanged, which means that the diagonal of M_1 must sum to 0.99. By assuming that evolution follows a Markov process it is simple to derive the mutation matrices for sequences separated by greater evolutionary distances. The M_n matrix, corresponding to a distance of n PAMs is equal to M_1^n. Finally the q_{ij} values can be derived with $q_{ij} = p_j M_{ij}$. By choosing different values of n for the mutation matrix we can estimate scoring matrices for sequences that are at varying evolutionary distances from one another. The correspondence between PAMs and the observed proportion of different residues is not one to one, therefore a distance of 250 PAMs corresponds to around only about 20% of identical residues where a distance of 180 PAMs corresponds to around 27% identical residues.[161,162] Therefore the PAM_{250} matrix, derived from M_{250}, is suited to align more distantly related proteins than the PAM_{180} for example. By changing the mathematical model underlying the estimate of mutation probabilities, PAM-like matrices[163] were later developed based on the same principles.

The other main type of substitution matrix is the BLOSUM matrix (Block Substitution matrix), developed in 1992.[164] Instead of using whole, closely-related, protein sequences like the PAM matrices, the values of q_{ij} were estimated on highly conserved segments, called *blocks*, across whole protein families. The q_{ij} values are then estimated as the number of time amino acids i and j are aligned divided by the number of total amino acid pairs in the alignment. Therefore q_{ij} is the observed frequency of the aligned pair of amino acids i and j in all the conserved blocks. As with PAM matrices, several BLOSUM matrices were constructed, designed for aligning sequences with different evolutionary distances. The BLOSUM62 matrix was estimated on blocks in aligned sequences that are at most 62% identical, BLOSUM80 on sequences that are at most 80% identical. Therefore, inversely to the PAM matrices, the higher the number of the BLOSUM matrix the more suited it is to align more closely related sequences.

PAM and BLOSUM matrices have fairly broad use-cases and are widely used in alignment. However, many other protein substitution models exist. Instead of using log-odds, some substitution models were developed by estimating scores with maximum-likelihood approaches.[165,166] Some matrices were developed with very specific usage conditions in mind, tailored to specific types of proteins like transmembrane,[167,168] disordered[169] or polar/non-polar[170] proteins. Some matrices were developed to align sequences from specific organisms like *P. falciparum*[171] *(responsible for malaria)* or HIV.[172] A substitution matrix was even developed in 2005 specifically for global rather than local alignment.[173]

This wealth of protein substitution matrices reflects the biological and evolutionary diversity of proteins, however substitution matrices for aligning DNA sequences are much less common. Some work has been done to derive matrices similar to PAM matrices from DNA alignments.[174] Codon substitution matrices[175,176] have been developed as well, although they are used in DNA sequence alignment, ultimately they use knowledge derived from protein alignments.

2.1.4. Dealing with gaps

In the NW and SW examples of Section 2.1.2, as with the simplistic match/mismatch costs, we used a very simple cost of insertions and deletions: any indel has a cost of -1. As was the case with substitutions, this does not reflect the biological reality very well.

In biology, when insertions or deletions occur it is more likely that the indel will span several nucleotides rather than just one.[177] This means that longer gap stretches are more likely than many individual gaps. For example, the two alignments below have the same number of matches, mismatches, and gaps. The second one is more likely since it is the result of a single insertion (or deletion) of `AGGT` rather than multiple independent indels.

```
AGGAGGTTCG      AGGAGGTTCG
A-G-G-T-CC      AGG----TCC
```

The first approach to take this into account was to try and optimize the gaps more generally[178] over the whole aligned sequence. However, even with dynamic programming, this has at best a time complexity of $\mathcal{O}(n^2m)$.[179] In 1982, Gotoh proposed affine gap costs.[180] With this model there are two separate costs associated to indels: 1) the gap open cost and 2) the gap extend cost. Usually the costs are set up so that opening a new gap is more costly than extending it, meaning that longer gap stretches are favored over many short indels. The other major advantage is that with Gotoh's algorithm time complexity is back down to $\mathcal{O}(nm)$. The algorithm was further refined by Altschul *et al.*.[181]

Over the years different types of gap costs were developed and tested like the logarithmic gap costs proposed by Waterman[182] and improved by Miller and Myers[183] which turned out to be less accurate than affine gap costs[184]). A bi-linear gap cost was also proposed to replace the affine cost,[185] with a breakpoint at gaps of length three, the size of a codon. As more and more sequence data became available, similarly to what happened with substitution matrices, empirical profile-based models derived from this data were developed.[186] Some of these penalties leverage structural information and context for proteins.[187,188] A context dependent gap penalty depending on the hydrophobicity of aligned residues is implemented in `Clustal X`,[189] one of the most widely used sequence aligners. Although quite complex and empirically derived, these profile-based penalties show limited improvement over the affine and bi-linear penalties.[190]

More recently, methodological and algorithmic developments have resulted in the WaveFront algorithm (WFA) for pairwise alignment.[191] This algorithm computes a NW alignment with affine gap costs with a much lower time complexity of $\mathcal{O}(ns)$, where s is the alignment score, reducing the quadratic relationship to sequence length to a linear one. This algorithm is also easily vectorizable and can take advantage of hardware acceleration, making its implementation run between 10 to 300 times faster than alternative methods depending on the testing context.[191]

2.2. How to speed up pairwise alignment ?

The NW and SW algorithms, as well as their improvements, are proven to be optimal.[192] However, when dealing with large sequences, which are more and more common, or when having to do many pairwise alignments, they become limiting due to their time and space complexity. In many cases, to get around these limitations, optimality is left aside in favor of heuristics and approximate methods speeding up alignment.

2.2.1. Changing the method

One of the early approaches to speed up alignment was to focus on speeding up the dynamic programming which is the time and space consuming step of the NW and SW algorithms. Bounded dynamic programming[193] is one such approach, in some works it is also called banded dynamic programming. By making the assumption that the majority of alignment operations are matches and mismatches instead of indels we can make a hypothesis about the alignment graph. Most probably, the path in the graph corresponding to the optimal alignment will be around the diagonal of the dynamic programming matrix, and scores far away from the diagonal are probably not needed. By making these assumptions a lot of the scores of the matrix do not need to be computed, speeding up the execution and leading to a sparse dynamic programming matrix (shown in Figure 2.3). This approach was used to speed up alignment early on in 1984.[194] The advantage of this method is that the optimal alignment can be found very efficiently. If there are many indels in the optimal alignment, this algorithm is not guaranteed to run faster than NW.

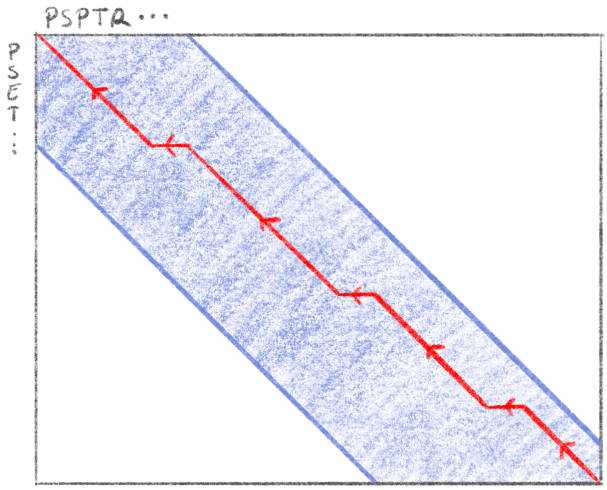


Figure 2.3.: **Bounded dynamic programming to speed up alignment.** The dynamic programming matrix is shown here, only values in the blue section are computed, speeding up the process. Here the optimal path in the alignment graph, shown in red, is included entirely in the bounds. Adapted from.[195]

More "exotic" methods have also been used successfully for sequence alignment. Fast Fourier Transform (FFT) are used in the `MAFFT` aligner[196] in order to quickly find homologous segments between two sequences. These homologous regions can be used as the basis for alignment. `MAFFT`, primarily a multiple sequence aligner *(c.f. Section 2.4 below)*, can also be used for pairwise alignment.

2.2.2. Seed and extend with data structures

In parallel to the development of new alignment algorithms, another way of substantially speeding up pairwise alignment is the so-called "seed and extend" method. This is based on the observation that a pairwise alignment most likely has several short subsequences that are almost identical in both sequences to align. These homologous subsequences, *seeds*, can be used to initialize an alignment that can be extended in both directions with dynamic programming until we have a suitable alignment.

This method can be used for 1) local alignment, where seeds indicate possible local matches which can be extended in local alignments; or 2) for global alignment where the seeds anchor the dynamic programming matrix, limiting the number of cells to fill out as shown in Figure 2.4. In both cases this approach follows the divide and conquer philosophy and extending seeds or filling out the dynamic programming matrix between anchors can be done independently and in parallel.

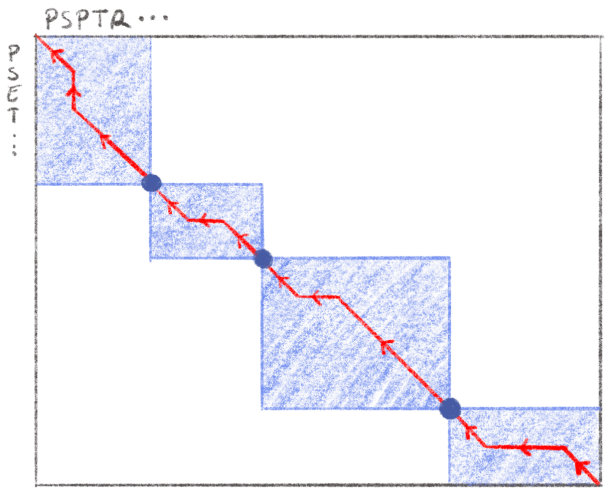


Figure 2.4.: **Divide and conquer to speed up alignment.**
Here anchors are used to speed up alignment. Anchors are shown as dark blue dots in the dynamic programing matrix. Only values in blocks between anchors, shown in blue, need to be computed. The majority of the matrix can be left empty. The optimal path in the resulting alignment graph must go through each anchor and is shown in red. Adapted from.[195]

This type of approach can also be used for many-to-one local alignments: either trying to find homologies between a query sequence and a database of sequences, or to find several local alignments in a large reference sequence like in read-mapping (see Section 2.3.1). In these many-to-one scenarios it is useful to index seeds in data structures that allow rapid querying and compact storage. This general framework

has proven to be quite flexible with many different ways to pick seeds[197] and many different data structures to index them.[198]

2.2.2.1. k-mers and hash tables

2.2.2.1.1. The BLAST algorithm One of the early methods for very quick heuristic alignment is the Basic Local Alignment Search Tool: `BLAST`.[199] It is widely used to this day to find homologous sequences in large databases and, as such, is one of the most cited papers of all time with over 100,000 citations. It is available as a web service hosted by the NCBI (https://blast.ncbi.nlm.nih.gov/Blast.cgi). Over the year many different versions for different use cases have been developed like `BLASTP` for protein sequences or `BLASTN` and `MEGABLAST` for nucleic acid sequences.

In our description of the `BLAST` algorithm, we will have a *target* and a *query* sequence that we wish to align.

1. For both sequences we build a hash table that uses subsequences of length k, called k-*mers*, as keys and their position in the whole sequence as values.
2. The hash tables are then scanned to check for exact matches between k-mers in the target and query sequences, called *hits*.
3. The positions of the hits in the target and query sequences are used to seed a candidate local alignment.
4. The candidate local alignments are extended in both directions from the seed with the SW algorithm. If the alignment score reaches a value under a specified threshold, the alignment stops and the candidate is discarded.

By selecting the right size k of the seeds (by default 11 when aligning nucleotides, 3 when aligning amino acids) as well as the alignment score threshold, one can adjust the sensitivity of the method at the cost of runtime.

It might not seem very useful to pre-compute the target hash-table for a single target. However, in practice `BLAST` is used to find local alignments between a query sequence and a very large number of target sequences; databases hosted by NCBI have hundreds of millions of target sequences (https://ftp.ncbi.nlm.nih.gov/blast/db/), at these scales pre-computing the target database saves an enormous amount of time.

Over time, several improvements have been developed for `BLAST`. `PSI-BLAST`[200] iteratively refines the alignments, Gapped `BLAST`[200]and `BLASTZ`[201] use spaced seeds, introduced in the PatternHunter method,[202] corresponding to seeds where not all characters match, increasing sensitivity. By sorting the target sequences it is possible to stop earlier and gain some speed as well.[203] The `Diamond` aligner[204] increase alignment speed by using double indexing and thus leveraging CPU cache and reducing time waiting for memory or disk access, improving alignment speed up to 360-fold over `BLAST` in later version.[205]

`FASTA`,[206] an improvement on `FASTP`,[207] is another method for local alignment, similar to `BLAST`. k-mers for the target and query sequence are indexed in a hash table and hits are found between the two sequences. The k-mers used in the `FASTA` tool are usually shorter than for `BLAST`, so instead of initializing an alignment at a single hit, `FASTA` identifies regions in both sequences that have a high density of hits, keeping the best 10. These regions are then scored using matrices discussed in Section 2.1.3 and high scoring regions are combined to build an approximate alignment. An optimal version of this alignment is then computed using the SW algorithm with banded dynamic programming.

Both `FASTA` and `BLAST` are very fast. It only takes a couple of seconds to find approximate local alignments between a hundred query sequences[208] in a database of over eighty million target sequences.[209] Attempting this task with standard SW or NW algorithms would be much slower[210] but would yield more sensitive, optimal alignments.[211]

2.2.2.1.2. Other algorithms One of the problems with such an approach is the size of the index, indeed storing all the k-mers of a length n sequence would require a maximum of $(n-k+1)\cdot k$ characters as the hash table keys, if all k-mers are distinct. This space constraint is acceptable for very large scale homology search tools on hosted web services such as NCBI BLAST, on a personal computer this can easily exceed memory capacity. Storing the hash table on disk has drastic consequences on query times, therefore methods to reduce the storage needs of these data structures were developed.

One of the ways to make everything fit in memory is to not store all k-mers. One way is through the use of so-called *minimizers*, introduced independently in 2003[212] and 2004.[213] Given a window of w consecutive k-mers and an ordering, a (w, k) minimizer is the "smallest" k-mer in the window w.r.t. the chosen ordering. Let us consider the following window of 3-mers with $w = 4$: `TGACAT`, yielding the following 3-mers: `TGA`, `GAC`, `ACA`, `CAT`. Following a simple ordering, such as the lexicographical ordering (i.e. alphabetical order), the "smallest" 3-mer, and our $(4, 3)$ minimizer, would be `ACA`, and only this one would be sampled and added to our hash table. Minimizers have interesting properties: adjacent windows often share a minimizer (see Figure 2.5) and if two strings have a $w - k + 1$ sequence in common then they are guaranteed to share a (w, k) minimizer.[213] These properties make minimizers very useful for the seed and extend alignment strategy and they are used in several aligners such as `minimap`[214] and `minimap2`,[119] `mashmap2`[215] and `winnowmap`.[120]

While the lexicographical ordering is easy to conceptualize, and the one proposed initially by Roberts *et al.*, it has an undesirable characteristic: it tends to select simpler k-mers with repeated `A` at the beginning. As discussed in Section 1.4.2, repeated stretches of nucleotides are prone to sequencing errors and as such are not ideal for seeding alignments. Furthermore, when the window shifts k-mers at the

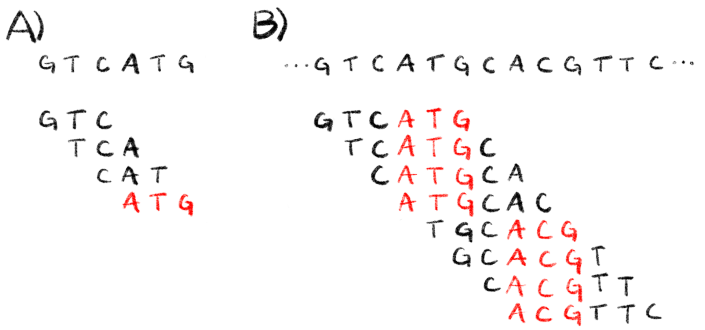


Figure 2.5.: ***k*-mer minimizers in action.**
A) The 3-mers are shown under a window of size $w = 4$ k-mers. The $(4, 3)$ minimizer according to the lexicographical ordering is highlighted in red. **B)** All the $w = 4$ windows of 3-mers are shown underneath the sequence. $(4, 3)$ minimizers of each window are highlighted in red. Here both 3-mer minimizer are shared by 4 windows. Adapted from.[213]

beginning of successive are likely to be selected as minimizers without being shared between windows, meaning that we sample more k-mers than needed. Roberts *et al.* proposed an alternative ordering based on nucleotide frequencies,[213] however this is also not ideal. Different orderings have been studied and those based on universal hitting sets,[216] or random orderings (such as the ones defined by a hash function) have more desirable properties than the lexicographical ordering.[217] A minimizer ordering based on frequency of appearance of k-mers has also been shown to provide well-balanced partitioning of k-mer sets.[218]

Over the years more strategies have been developed to sample k-mers and reduce the data structure size for efficient sequence alignment, such as syncmers,[219] strobemers[220] or a combination of both.[221] These novel seed sampling strategies allow for sparser seed sampling, smaller data structures and therefore faster alignment software.

2.2.2.2. Exact matches and suffix trees

While k-mer seeds have shown success it is not the only way to implement a seed and extend alignment method. The other way to seed alignments is through maximal exact matches (MEMs) which is the longest possible exact match between two sequences. MEMs can be found with data structures like suffix trees,[222] suffix arrays[223,224] or FM indices.[225]

Suffix trees have long been used for pattern matching applications,[129] the `AVID` aligner[226] uses them to find maximal exact matches between two sequences to anchor

a global alignment. MUMmer2[227] uses suffix trees to find unique maximal unique matches (MUMs) to anchor alignments.

Suffix trees, although very useful, have quadratic space complexity w.r.t. to the length of the indexed sequence.[129] This is fine for small bacterial or viral genomes. However, in the age of whole genome sequencing and the human genome project, it is inadequate. Therefore some aligners have switched data structures to use suffix arrays. In fact, it is possible to replace suffix trees with these more space efficient suffix arrays in any algorithm.[228] Newer versions of MUMmer[229] have made this choice and now use suffix arrays for improved performance. It is important to note that compact versions of suffix trees have been created that are also linear in space to sequence length,[230] however in practice suffix arrays take up less space for comparable query times.[223]

Finally another data structure that is widely used is the so-called FM index proposed in 2000[225] and based on the Burrows-Wheeler transform.[231] The FM index is very memory efficient,[232] but this comes at the cost of some efficiency in index lookup operations, although some work has been done to improve this.[233] As such, FM-indices have been used in many aligners such as BWT-SW,[234] BWA[235] and BWA-SW,[236] BWA-MEM,[237] CUSHAW[238] or bowtie2.[239]

The seed and extend paradigm has been very useful in the field of genomics to deal with the scale of data and keep up with sequencing technologies. Some newer alignment algorithms like the WFA algorithm mentioned above, have even been used in such a context.[240]

Finally, some methodological development have been aimed towards improving alignment sensitivity instead of speed. One of these methods, fairly well studied in general, and in the context of alignment, are hidden markov models (HMMs). In certain circumstances PairHMMs, HMMs used for pairwise alignment, can be mathematically equivalent to NW.[241] HMMs have been used for sequence alignment in many software tools like HHsearch,[242] HMMer[243] or MCALIGN2[244] which is used to efficiently search for alignments in large databases of sequences.

2.3. The specificities of read-mapping

Read-mapping is a special case of pairwise alignment and the focus of Chapter 3, it stands to reason that we use this section to explain the stakes and challenges of the read-mapping task.

2.3.1. What is read-mapping ?

Read-mapping, or sometimes read-alignment is the process of comparing a sequencing read to a reference sequence and finding the region in the reference homologous

to the read. Sometimes, mappers only output the position where this region starts in the reference but more often than not, they output local or semi-global alignments between the reads and the reference. In *semi-global alignment*, two sequences are globally aligned but indels at the end and beginning of each sequence are not penalized, this can be useful to detect overlap between two sequences, or align two sequences of very different sizes.

Read-mapping is often the first step of many bioinformatics analysis pipelines, and as such is often crucial. Therefore it makes sense that this is a very active field with many reviews[245–249] and some benchmarking procedures[250] to compare tools.

From a technical and algorithmic standpoint, the task of mapping many sequencing reads to a single reference lends itself very well to the "divide and conquer" approach presented in Section 2.2.2. Indexing the reference beforehand and using this index as a database to align can lead to substantial execution speed gains. As a matter of fact, many of the aligners presented in Section 2.2.2 are actually read-mappers that can also do pairwise alignment. As such most implement the seed-and-extend paradigm with hash-tables like `minimap2`;[119] FM-indices like `BWT-SW`,[234] `bowtie2`,[239] `BWA`,[235] `BWA-SW`,[236] `BWA-MEM`[237] and `CUSHAW`;[238] or even other divide and conquer approaches like `Kart`.[251] As sequencing technologies yield longer and more numerous reads, these heuristics become more important if we wish to be able to analyze this data. This can be partly mitigated through hardware acceleration.[252–255]

2.3.2. Challenges of read-mapping

Read-mapping, as one might expect, is no easy task. The length of recent sequencing reads and their numbers are of course challenging, but algorithmic tricks described above can help. There are other aspects of sequencing data that make read-mapping as hard as it is.

Sequencing technologies, although they have improved over time, can still make errors, and these errors can lower the homology between reads and reference, making mapping harder.[129] This is particularly true of long reads where the error rate is higher. To mitigate that some specific long-read mappers take these errors into account when aligning a read to the reference. Some mappers are tied to a specific sequencing technology like BLASR[256] or `lordFAST`[257] for PacBIO reads, and `GraphMap`[258] for ONT. Some however, like `NGMLR`,[259] `MashMap`[260] or `DuploMap`,[261] are technology agnostic and can work with any type of long-read. This might not be needed forever though as sequencing accuracy is growing with every new generation of sequencers. Since homopolymer-linked indels are still common in long-read sequencing (cf. Section 1.4.2) many modern read-mappers, designed to work with long reads, include some option to use homopolymer compression (c.f. Section 1.4.3.2).

While the technology producing reads can complicate the read-mapping tasks, some regions of the genome are intrinsically harder to map to. This is particularly true of

repetitive regions like telomeres or centromeres.[249] Repetitive regions mean a lot of potential homologous regions between a read and the reference, producing a lot of seed hits, increasing the runtime of the aligners and lowering the overall confidence in read-placement. Some tools have been developed specifically to deal with such regions. `Winnowmap`[120] and `Winnowmap2`,[262] assign a weight to k-mers that might be sampled as minimizers. By under-weighting frequently appearing k-mers they can improve performance in repetitive regions. `TandemMapper`[263] was designed to map long reads to the extra-long tandem repeats (ETRs) present in centromeric regions. It does not use minimizers, like `Winnowmap` it selects less frequent k-mers as potential seeds to deal with the repetitiveness and improve the mapping accuracy. Long reads are also much easier to map to repetitive regions since they can span over them, or overlap with more complex regions.[48,56]

Some challenges however are linked to implementation rather than sequencing data. Some efforts have been done to provide quality scores to mappings in order to easily assess their quality and therefore usefulness. This score, called *mapping quality*, is defined as $-10\log_{10}(p)$, usually rounded to the nearest integer, where p corresponds to the probability of the read being mismapped. It was introduced in the `MAQ` software[264] but has been implemented in many read-mappers like `BWA`,[235] `bowtie2`[239] or `minimap2`[119] since it was added as part of the widely-used `SAM` file format specification.[265]

While the mapping quality score is standardized, each read-mapper has a different way of estimating p, the mismap probability. This creates differences in the reported qualities: *e.g.* the maximum quality that `bowtie2` can assign is 42, `BWA's` is 37 and `minimap's` is 60.[266] This of course means that comparing mapping quality values between read-mappers is not necessarily meaningful. Furthermore in some cases this mapping quality is not very reflective of the alignment accuracy,[245] as such alternative approaches have been explored: through a new genome mappability score,[267] simulations[268] or even machine learning.[269]

In conclusion, as a crucial step in many bioinformatics pipelines, read-mapping is a markedly active field with a lot of work in increasing mapping accuracy and speeding up alignment. However, despite all this work, some challenges remain. Further improving mapping is possible and doing so could result in more accurate downstream analyses and avoid drawing some erroneous conclusions.

2.4. Multiple sequence alignment

Up until now we have only considered pairwise alignment where we want to find homologies between a pair of sequences. In many cases though, it is helpful to compare more than two sequences together, this is where *multiple sequence alignment* (MSA) steps-in. It is an essential task in many bioinformatics and comparative biology analyses.[270]

We saw earlier that with dynamic programming and algorithms like NW or SW it is possible to compute an optimal pairwise alignment. For MSA, the task of computing the optimal alignment is, unfortunately, NP-hard,[271,272] with an exponentially-growing time and space complexity in the number of sequences to align. Therefore, heuristics and approximations are needed from the start in order to get anything meaningful.

An early method, and easy to conceptualize, is the so-called center star alignment method.[195] In this approach, a single sequence is chosen to be the center sequence. After this each other sequence is aligned to the center sequence and the pairwise alignments are merged, conserving gaps that were inserted. The center sequence is often chosen to be as similar to the other sequences as possible. For this, all pairwise distances between sequences are needed implying a quadratic distance computation step. The pairwise alignments are independent so this approach is easy to parallelize. Some software, like `HAlign`[273] use center star alignment to produce MSAs. This method, however, is quite sensitive to the choice of the center sequence. Bad pairwise alignments can lower the accuracy of the overall MSA by conserving gaps.

2.4.1. Progressive alignment

One of the most widely used multiple sequence alignment approach is progressive alignment.[274] Similarly to the center star algorithm, the progressive algorithm reduces the MSA problem to independent pairwise alignments. The first step is to build a phylogenetic tree from the sequences to align, representing the evolutionary relationship between sequences, called the *guide tree*. Starting from the leaves, that correspond to single sequences, pairwise align the sequences and store the alignment (or *profile*) in the parent node. Going up from the leaves to the roots, align sequences together, then sequences to profiles if needed and finally profiles together, merging alignments as we progress up the tree. The final multiple sequence alignment is obtained when this process reaches the root. Profiles at inner nodes of the tree are aligned to each other to conserve gaps. A representation of this process is shown in Figure 2.6.

In many cases a matrix of pairwise distances is needed to construct the guide tree, if we choose the edit distance, $n(n-1)/2$ pairwise alignments are needed to get this matrix. With a large number of sequences, or long sequences this is not possible in a reasonable amount of time. Therefore, computing of distance matrices through alignment-free methods, usually based on k-mers, is often used as input to the tree building method.[275,276]

Tree reconstruction methods from the distance matrix like UPGMA[277] or neighbor-joining[278] can be quite time consuming when dealing with a large set of sequences. To counteract this, some multiple sequence aligners also use heuristic methods to

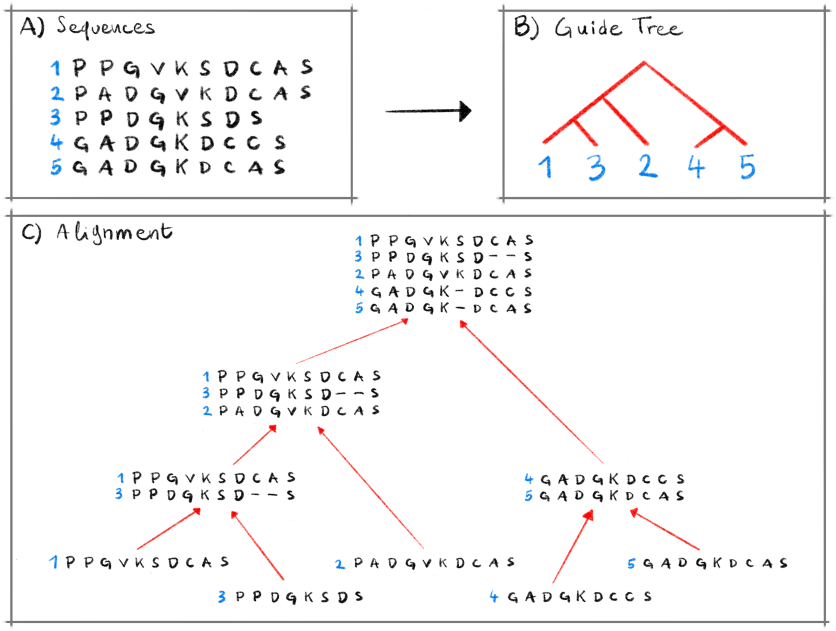


Figure 2.6.: **Overview of the progressive alignment process.**
A) sequences to align, **B)** guide tree constructed from distances between sequences in panel A, **C)** Alignment steps along the guide tree and resulting MSA at the root of the tree. Adapted from[144]

approximate a guide tree. `MAFFT`,[196] for example, uses `PartTree`[279] method to approximate the tree, and `clustal Omega`[280] uses an embedding method[281] to do so.

Although this method is a good heuristic as the guide tree can capture complex relationships between sequences, progressive alignment can still suffer from problems similar to center star alignment: mainly gap propagation. If an early alignment is erroneous and introduces spurious gaps, then these are propagated throughout the MSA. As it is said in the seminal progressive alignment paper: "once a gap, always a gap".[274] Iterative refinement of the MSA[270] was proposed as a solution to this problem. A possible approach is to recompute a guide tree from the alignment and run the whole progressive alignment procedure on the new guide tree, but this is very time consuming and is not practical with the large sequence sets available today. Therefore, the iterative procedure consists of taking an MSA obtained through progressive alignment and splitting it horizontally in two alignments of $n/2$ sequences each. In each half, the sites composed exclusively of gaps are removed and the two alignments merged through profile alignment. After the realignment of both halves, a scoring metric is computed and as long as this metric improves, repeat the previous steps. There are several of these metrics, the most commonly used is the probably the sum of pairs score[282] or its weighted variant.[283] There exist other scores like log-odds and correlation[284] or a consistency based score.[285]

Most of the widely used multiple sequence aligners some form of progressive alignment with iterative refinement: `T-Coffee`[286] which uses a consistency score for refinement, `MUSCLE`,[287,288] `MAFFT`,[196] `ProbCons`[289] which uses a formal HMM to compute consistency and the various `CLUSTAL` incarnations[280,290,291] which are some of the most cited papers of all times.

2.4.2. Other methods

While the progressive alignment algorithm has been at the root of some of the most widely used alignment software, other methods to produce MSAs have been explored over the years.

One common other method for creating multiple alignment, whether through profile-profile alignment or sequence-profile alignments are HMMs. Several tools HMMs to generate an alignment such as `HMMer`,[243] `MSAProbs`[292] or `COVID-align`.[293] In some cases, the HMM based approach has similar performance to `clustalW`.[294]

Other methods have focused on speeding up the dynamic programming part of aligning multiple sequences. This can be done using simulated annealing,[295–297] which can also be used to speed up HMM training.[294] Genetic algorithms have also been used to construct MSAs,[298] increasing the speed at which this is possible.[299] Several tools use genetic algorithms like `VGDA`,[300] `GAPAM`[301] and `SAGA`.[302]

With the recent focus on SARS-CoV 2, some specific multiple sequence aligners have been developed to create very large multiple sequence alignments. They often take advantage of the fact that this virus mutates quite slowly meaning that most of the available sequences have very high homology. Furthermore, as the epidemic was tracked in near real time since its beginning, we know the original sequence at the root of the pandemic. Leveraging this knowledge, it is possible to build a profile from aligning new sequences to the ancestral sequence and aligning new sequences to this profile using HMMs like what is done in `COVID-align`.[293] The `NextAlign`[303] software even forgoes aligning to a profile and creates massive MSAs (millions of sequences) by aligning new sequences to the ancestral sequence using banded SW alignment, the gap penalties are enriched with biological knowledge and dependent on the position within the sequence.

Recently, Garriga *et al.* introduced the regressive alignment method,[304] where instead of traversing a guide tree from leaf to root, it goes the other way, aligning the more distant sequences first before merging MSAs. Using this approach, they managed to create an MSA of 1.4 million sequences with improved accuracy over progressive methods.

Since multiple sequence alignments are so useful in comparative biology, and that there is such a vast array of methods to construct them it stands to reason that are

many resources to help practitioners make their choice. There are many reviews and benchmarking datasets and procedures to do so.[305–309]

2.5. Conclusion

Sequence alignment, multiple or pairwise, is a fundamental part of the bioinformatician's toolkit. Comparing sequences and finding homologies is at the root of many fields because of the wealth of evolutionary information contained in alignments. As such it is paramount to have the best possible sequence alignments in any situation.

As we have seen now, although we have methods guaranteed to give us optimal pairwise and multiple sequence alignments, they are not practically useful for dealing with sequences at today's scale. Therefore, most sequence aligners rely on, sometimes numerous, heuristics and approximations. From substitution models to seeding techniques, all these are not necessarily reflective of the biological reality contained within the sequences to align. Each of these heuristics or models is a step where biases and approximations can happen, building up along and over sequences. Therefore, there must be room for improvement.

Having methods that are both fast and accurate are now more necessary than ever with the ever growing scale and number of publicly available sequences. Furthermore, in the "age of pandemics", accurate alignment methods are indispensable to track and keep an eye on disease spread across the globe, in real-time.

3. Contribution 1: Improving Read Alignment by Exploring a Sequence Transformation Space

Recall that, when using long-read sequencing technologies, sequencing errors are more frequent than when using short-read sequencing. The most common of these sequencing errors are linked to homopolymers (1.4.2). In read-mapping analyses, a short sequence is globally aligned to a much longer reference sequence. Mapping long-reads can help bridge some gaps in knowledge and solve problems impossible to solve with shorter reads, however sequencing errors complicate an already complicated task (2.3). Homopolymer compression (1.4.3.2) has been successfully used to mitigate some of the effects of these errors and improve long-read mapping analyses. There might, however, be rooom for improvement and alternative sequence transformation procedures that improve long-read mapping more than homopolymer compression.

This chapter was written as an article titled: **"Mapping-friendly sequence reductions: going beyond homopolymer compression"**. It is currently in press for the iScience proceedings of the RECOMB-SEQ 2022 conference and is presented as is, without any modification from the submitted version. The author list, complete with affiliations is given below:

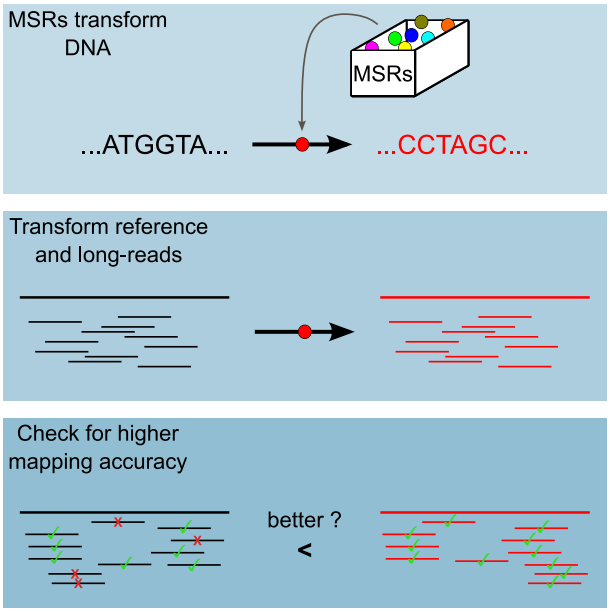
Luc Blassel[1,2*], Paul Medvedev[3,4,5], Rayan Chikhi[1]

1 Sequence Bioinformatics, Department of Computational Biology, Institut Pasteur, Paris, France
2 Sorbonne Université, Collège doctoral, Paris, France
3 Department of Computer Science and Engineering, Pennsylvania State University, University Park, Pennsylvania, United States of America
4 Department of Biochemistry and Molecular Biology, Pennsylvania State University, University Park, Pennsylvania, United States of America
5 Center for Computational Biology and Bioinformatics, Pennsylvania State University, University Park, Pennsylvania, United States of America

Highlights

- Mapping-friendly sequence reductions (MSRs) are functions that transform DNA sequences.
- They are a generalization of the concept of homopolymer compression.
- We show that some well-chosen MSRs enable more accurate long read mapping.

Graphical abstract



Abstract

Sequencing errors continue to pose algorithmic challenges to methods working with sequencing data. One of the simplest and most prevalent techniques for ameliorating the detrimental effects of homopolymer expansion/contraction errors present in long reads is homopolymer compression. It collapses runs of repeated nucleotides, to remove some sequencing errors and improve mapping sensitivity. Though our intuitive understanding justifies why homopolymer compression works, it in no way implies that it is the best transformation that can be done. In this paper, we explore if there are transformations that can be applied in the same pre-processing manner as homopolymer compression that would achieve better alignment sensitivity. We introduce a more general framework than homopolymer compression,

called mapping-friendly sequence reductions. We transform the reference and the reads using these reductions and then apply an alignment algorithm. We demonstrate that some mapping-friendly sequence reductions lead to improved mapping accuracy, outperforming homopolymer compression.

3.1. Introduction

Sequencing errors continue to pose algorithmic challenges to methods working with read data. In short-read technologies, these tend to be substitution errors, but in long reads, these tend to be short insertions and deletions; most common are expansions or contractions of homopolymers (i.e. reporting 3 As instead of 4).[101] Many algorithmic problems, such as alignment, become trivial if not for sequencing errors.[129] Error correction can often decrease the error rate but does not eliminate all errors. Most tools therefore incorporate the uncertainty caused by errors into their underlying algorithms. The higher the error rate, the more detrimental its effect on algorithm speed, memory, and accuracy. While the sequencing error rate of any given technology tends to decrease over time, new technologies entering the market typically have high error rates (e.g. Oxford Nanopore Technologies). Finding better ways to cope with sequencing error therefore remains a top priority in bioinformatics.

One of the simplest and most prevalent techniques for ameliorating the detrimental effects of homopolymer expansion/contraction errors is *homopolymer compression* (abbreviated HPC). HPC simply transforms runs of the same nucleotide within a sequence into a single occurrence of that nucleotide. For example, HPC applied to the sequence AAAGGTTA yields the sequence AGTA. To use HPC in an alignment algorithm, one first compresses the reads and the reference, then aligns each compressed read to the compressed reference, and finally reports all alignment locations, converted into the coordinate system of the uncompressed reference. HPC effectively removes homopolymer expansion/contraction errors from the downstream algorithm. Though there is a trade-off with specificity of the alignment (e.g. some of the compressed alignments may not correspond to true alignments) the improvement in mapping sensitivity usually outweighs it.[119]

The first use of HPC that we are aware of was in 2008 as a pre-processing step for 454 pyrosequencing data in the Celera assembler.[115] It is used by a wide range of error-correction algorithms, e.g. for 454 data,[310] PacBio data,[117] and Oxford Nanopore data.[311] HPC is used in alignment, e.g. by the widely used minimap2 aligner.[119] HPC is also used in long-read assembly, e.g. HiCanu,[112] SMARTdenovo,[312] or mdBG.[113] HPC is also used for clustering transcriptome reads according to gene family of origin.[116] Overall, HPC has been widely used, with demonstrated benefits.

Though our intuitive understanding justifies why HPC works, it in no way implies that it is the best transformation that can be done. Are there transformations

that can be applied in the same pre-processing way as HPC that would achieve better alignment sensitivity? In this work, we define a more general notion which we call *mapping-friendly sequence reductions*. In order to efficiently explore the performance of all reductions, we identify two heuristics to reduce the search space of reductions. We then identify a number of mapping-friendly sequence reductions which are likely to yield better mapping performance than HPC. We evaluate them using two mappers (`minimap2` and `winnowmap2`) on three simulated datasets (whole human genome, human centromere, and whole *Drosophila* genome). We show that some of these functions provide vastly superior performance in terms of correctly placing high mapping quality reads, compared to either HPC or using raw reads. For example, one function decreased the mapping error rate of `minimap2` by an order of magnitude over the entire human genome, keeping an identical fraction of reads mapped.

We also evaluate whether HPC sensitivity gains continue to outweigh the specificity cost with the advent of telomere-to-telomere assemblies.[4] These contain many more low-complexity and/or repeated regions such as centromeres and telomeres. HPC may increase mapping ambiguity in these regions by removing small, distinguishing, differences between repeat instances. Indeed, we find that neither HPC nor our mapping-friendly sequence reductions perform better than mapping raw reads on centromeres, hinting at the importance of preserving all sequence information in repeated regions.

3.2. Results

3.2.1. Streaming sequence reductions

We wish to extend the notion of homopolymer compression to a more general function while maintaining its simplicity. What makes HPC simple is that it can be done in a streaming fashion over the sequence while maintaining only a local context. The algorithm can be viewed simply as scanning a string from left to right and, at each new character, outputting that character if and only if it is different from the previous character. In order to prepare for generalizing this algorithm, let us define a function $g^{\text{HPC}} : \Sigma^2 \to \Sigma \cup \{\varepsilon\}$ where Σ is the DNA alphabet, ε is the empty character, and

$$g^{\text{HPC}}(x_1 \cdot x_2) = \begin{cases} x_2 & \text{if } x_1 \neq x_2 \\ \varepsilon & \text{if } x_1 = x_2 \end{cases}$$

Now, we can view HPC as sliding a window of size 2 over the sequence and at each new window, applying g^{HPC} to the window and concatenating the output to the

growing compressed string. Formally, let x be a string, which we index starting from 1. Then, the HPC transformation is defined as

$$f(x) = x[1, \ell - 1] \cdot g(x[1, \ell]) \cdot g(x[2, \ell + 1]) \cdots g(x[|x| - \ell + 1, |x|]) \tag{3.1}$$

where $\ell = 2$ and $g = g^{\text{HPC}}$. In other words, f is the concatenation of the first $\ell - 1$ characters of x and the sequence of outputs of g applied to a sliding window of length ℓ over x. The core of the transformation is given by g and the size of the context ℓ, and f is simply the wrapper for g so that the transformation can be applied to arbitrary length strings.

With this view in mind, we can generalize HPC while keeping its simplicity by 1) considering different functions g that can be plugged into Equation (3.1) increasing the context that g uses (i.e. setting $\ell > 2$). Formally, for a given alphabet Σ and a context size ℓ, a function T mapping strings to strings is said to be an *order-ℓ Streaming sequence reduction* (abbreviated *SSR*) if there exists some $g : \Sigma^\ell \to \Sigma \cup \{\varepsilon\}$ such that $T = f$.

Figure 3.1A shows how an SSR can be visualized as a directed graph. Observe that an order-ℓ SSR is defined by a mapping between $|\Sigma|^\ell$ inputs and $|\Sigma| + 1$ outputs. For example, for $\ell = 2$, there are $n = 16$ inputs and $k = 5$ outputs. Figure 3.1B visualizes HPC in this way.

Since we aim to use SSRs in the context of sequencing data, we need to place additional restrictions on how they handle reverse complements. For example, given two strings x (e.g. a read) and y (e.g. a substring of the reference), a mapper might check if $x = RC(y)$. When strings are pre-processed using an SSR f, it will end up checking if $f(x) = RC(f(y))$. However, $x = RC(y)$ only implies that $f(x) = f(RC(y))$. In order to have it also imply that $f(x) = RC(f(y))$, we need f to be commutative with RC, i.e. applying SSR then RC needs to be equivalent to applying RC then SSR. We say that f is *RC-insensitive* if for all x, $f(RC(x)) = RC(f(x))$. Observe that HPC is RC-insensitive.

3.2.2. Restricting the space of streaming sequence reductions

To discover SSRs that improve mapping performance, our strategy is to put them all to the test by evaluating the results of an actual mapping software over a simulated test dataset reduced by each SSR. However, even with only 16 inputs and 5 outputs, the number of possible g mappings for order-2 SSRs is $5^{16} \approx 1.5 \cdot 10^{11}$, which is prohibitive to enumerate. In this section, we describe two ideas for reducing the space of SSRs that we will test. In subsection 3.2.2.1, we show how the restriction to RC-insensitive mappings can be used to reduce the search space. In subsection 3.2.2.2, we exploit the natural symmetry that arises due to Watson-Crick complements to further restrict the search space.

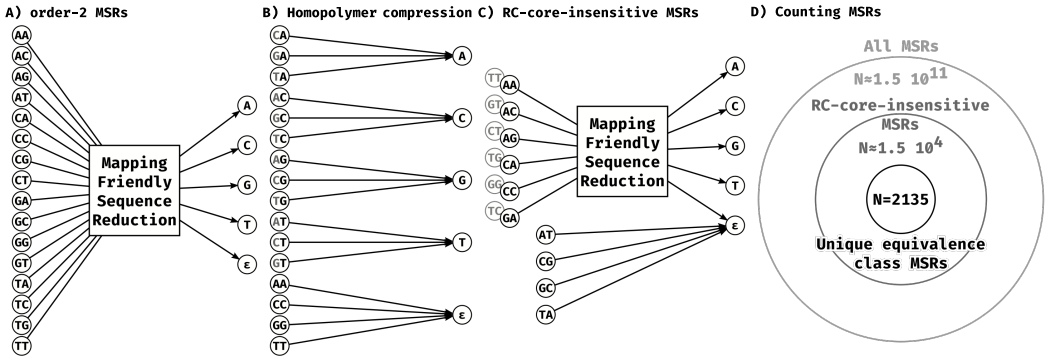


Figure 3.1.: **Representing and counting Streaming sequence reductions.** **A**: General representation of an order-2 Streaming sequence reduction as a mapping of 16 input dinucleotides, to the 4 nucleotide outputs and the empty character ε. **B**: Homopolymer compression is an order-2 SSR. All dinucleotides except those that contain the same nucleotide twice map to the second nucleotide of the pair. The 4 dinucleotides that are the two same nucleotides map to the empty character ε. **C**: Our RC-core-insensitive order-2 SSRs are mappings of the 6 representative dinucleotide inputs to the 4 nucleotide outputs and the empty character ε. The 4 dinucleotides that are their own reverse complement are always mapped to ε. The remaining 6 dinucleotides are mapped to the complement of the mapped output of the reverse complement dinucleotide input. For example, if AA is mapped to C, then TT (the reverse complement of AA) will be mapped to G (the complement of C). **D**: Number of possible SSR mappings under the different restrictions presented in the main text. All mappings from 16 dinucleotide inputs to 5 outputs (as in panel A) are represented by the outermost circle. All RC-core-insensitive mappings (as in panel C) are represented by the medium circle. All RC-core-insensitive mappings with only one representative of each equivalence class are represented by the innermost circle.

These restrictions reduce the number of order-2 SSRs to only , making it feasible to test all of them. Figure 3.1D shows an overview of our restriction process.

3.2.2.1. Reverse complement-core-insensitive streaming sequence reductions

Consider an SSR defined by a function g, as in Equation (3.1). Throughout this paper we will consider SSRs that have a related but weaker property than RC-insensitive. We say that an SSR is *RC-core-insensitive* if the function g that defines it has the property that for every ℓ-mer x and its reverse complement y, we have that either $g(x)$ is the reverse complement of $g(y)$ or $g(x) = g(y) = \varepsilon$. We will restrict our SSR search space to RC-core-insensitive reductions in order to reduce

the number of SSRs we will need to test.

Let us consider what this means for the case of $\ell = 2$, which will be the focal point of our experimental analysis. There are 16 ℓ-mers(i.e. dinucleotides) in total. Four of them are their own reverse complement: AT, TA, GC, CG. The RC-core-insensitive restriction forces g to map each of these to ε, since a single nucleotide output cannot be its own reverse complement. This leaves 12 ℓ-mers, which can be broken down into 6 pairs of reverse complements. For each pair, we can order them in lexicographical order and write them as $(AA, TT), (AC, GT), (AG, CT), (CA, TG), (CC, GG)$, and (GA, TC). Defining g can then be done by assigning an output nucleotide to the first ℓ-mer in each of these pairs (Figure 3.1C). For example, we can define an SSR by assigning $g(AA) = C$, $g(AC) = C$, $g(AG) = A$, $g(CA) = A$, $g(CC) = T$, and $g(GA) = G$. As an example, let us apply the corresponding SSR to an example read r:

$$\begin{aligned} r &= \text{TAAGTTGA} & f(RC(r)) &= \text{TCACCTG} \\ f(r) &= \text{TCAGGTG} & RC(f(r)) &= \quad \text{CACCTGA} \\ RC(r) &= \text{TCAACTTA} \end{aligned}$$

Observe that the first $\ell-1$ nucleotides of r (shown in red) are copied as-is, since we do not apply g on them (as per Equation (3.1)). As we see in this example, this implies that $f(RC(r))$ is not necessarily equal to $RC(f(r))$; thus an RC-core-insensitive SSR is not necessarily an RC-insensitive SSR. However, an RC-core-insensitive SSR has the property that for all strings r, we have $f(RC(r))[\ell, |r|]) = RC(f(r))[1, |r|-\ell+1]$. In other words, if we drop the $\ell-1$ prefix of $f(RC(r))$ and the $\ell-1$ suffix of $RC(f(r))$, then the two strings are equal. Though we no longer have the strict RC-insensitive property, this new property suffices for the purpose of mapping long reads. Since the length of the read sequences will be much greater than ℓ (in our results we will only use $\ell = 2$), having a mismatch in the first or last nucleotide will be practically inconsequential.

It is important to note though that there may be other RC-insensitive functions not generated by this construction. For instance, HPC cannot be derived using this method (as it does not map the di-nucleotides AT,TA,GC and CG to ε), and yet it is RC-insensitive.

We can count the number of RC-core-insensitive SSRs. Let us define $i(\ell)$ the number of input assignments necessary to fully determine the RC-core-insensitive SSR; one can think of this as the degrees-of-freedom in choosing g. As we showed, for $\ell = 2$, we have $i(\ell) = 6$. The number of RC-core-insensitive SSRs is then $5^{i(\ell)}$. Therefore, for $\ell = 2$, instead of 5^{16} possible mappings we have at most $5^6 \approx 1.5 \cdot 10^4$ RC-core-insensitive mappings (Figure 3.1D). For an odd $\ell > 2$, there are no ℓ-mers that are their own reverse complements, hence $i(\ell) = 4^\ell/2$. If ℓ is even then there

are $4^{\ell/2}$ inputs that are their own reverse complements (i.e. we take all possible sequences of length $\ell/2$ and reconstruct the other half with reverse complements). Thus, $i(\ell) = (4^{\ell} - 4^{\ell/2})/2$.

3.2.2.2. Equivalence classes of SSRs

Non mapping-related preliminary tests led us to hypothesize that swapping $A \leftrightarrow T$ and/or $C \leftrightarrow G$, as well as swapping the whole A/T pair with the C/G pair in the SSR outputs would have a negligible effect on performance. In other words, we could exchange the letters of the output in a way that preserves the Watson-Crick complementary relation. Intuitively, this can be due to the symmetry induced by reverse complements in nucleic acid strands, though we do not have a more rigorous explanation for this effect. In this section, we will formalize this observation by defining the notion of SSR equivalence. This will reduce the space of SSRs that we will need to consider by allowing us to evaluate only one SSR from each equivalence class.

Consider an RC-core-insensitive SSR defined by a function g, as in Equation (3.1). An ℓ-mer is canonical if it is the not lexicographically larger than its reverse complement. Let I be the set of all ℓ-mers that are canonical. Such an SSR's *dimension* k is the number of distinct nucleotides that can be output by g on inputs from I (not counting ε). The dimension can range from 1 to 4. Next, observe that g maps all elements of I to one of $k+1$ values (i.e. $\Sigma \cup \varepsilon$). The output of g on ℓ-mers not in I is determined by its output on ℓ-mers in I, since we assume the SSR is RC-core-insensitive. We can therefore view it as a partition of I into $k+1$ sets S_0, ..., S_k, and then having a function t that is an injection from $\{1, \dots, k\}$ to Σ that assigns an output letter to each partition. Further, we permanently assign the output letter for S_0 to be ε. Note that while S_0 could be empty, $S_1, \dots, S_k$ cannot be empty by definition of dimension. For example, the SSR used in Section 3.2.2.1 has dimension four and corresponds to the partition $S_0 = \{\}, S_1 = \{AG, CA\}$, $S_2 = \{CC\}$, $S_3 = \{AA, AC\}$, $S_4 = \{GA\}$, and to the injection $t(1) = A$, $t(2) = T$, $t(3) = C$, and $t(4) = G$.

Let $\textsc{IsComp}(x, y)$ be a function that returns true if two nucleotides $x, y \in \Sigma \cup \{\varepsilon\}$ are Watson-Crick complements, and false otherwise. Consider two SSRs of dimension k defined by $S_0, \dots, S_k, t$ and $S'_0, , S'_k, t'$, respectively. We say that they are equivalent iff all the following conditions are met:

- $S_0 = S'_0$,
- there exists a permutation π of $\{1, \dots, k\}$ such that for all $1 \leq i \leq k$, we have $S_i = S'_{\pi(i)}$,
- for all $1 \leq i < j \leq k$, we have $\textsc{IsComp}(t(i), t(j)) = \textsc{IsComp}(t'(\pi(i)), t'(\pi(j)))$.

One can verify that this definition is indeed an equivalence relation, i.e. it is reflexive, symmetric, and transitive. Therefore, we can partition the set of all SSRs into equivalence classes based on this equivalence relation. One caveat is that a single SSR defined by a function g may correspond to multiple SSRs of the form $S_0, \dots, S_k, t$. However, these multiple SSRs are equivalent, hence the resulting equivalence classes are not affected. Furthermore, we can assume that there is some rule to pick one representative SSR for its equivalence class; the rule itself does not matter in our case.

Figure 3.2 shows the equivalence classes for $\ell = 2$, for a fixed partition. An equivalence class can be defined by which pair of classes S_i and S_j have complementary outputs under t and t'. Let us define $o(k)$ as the number of equivalence classes for a given partition and a given k. Then Figure 3.2 shows that $o(1) = 1$, $o(2) = 2$ and $o(3) = o(4) = 3$. There are thus only 9 equivalence classes for a given partition.

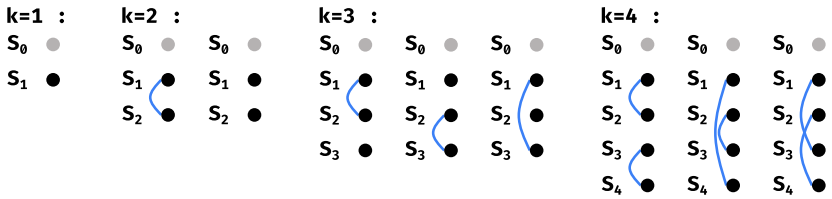


Figure 3.2.: **SSR equivalence classes for a fixed partition of the inputs.** S_0 is always assigned ε, so it is represented by a gray node. A blue link between S_i and an S_j denotes that IsComp$(t(i), t(j))$ = true. The equivalence classes are determined by the Watson-Crick complementary relationships between the rest of the parts, i.e. by all the possible ways to draw the blue links.

3.2.2.3. Counting the number of restricted SSRs

In this section, we derive a formula for the number of restricted SSRs, i.e. SSRs that are RC-core-insensitive and that are representative for their equivalence class. Consider the class of RC-core-insensitive SSRs with dimension k. In subsection 3.2.2.1, we derived that the degrees-of-freedom in assigning ℓ-mers to an output is $i(\ell) = 4^\ell/2$ if ℓ is odd and $i(\ell) = (4^\ell - 4^{\ell/2})/2$ if ℓ is even. Let $C(\ell, k)$ be the number of ways that $i(\ell)$ ℓ-mers can be partitioned into $k+1$ sets $S_0, \dots, S_k$, with $S_1, \dots, S_k$ required to be non-empty. Then, in subsection 3.2.2.2, we have derived $o(k)$, the number of SSR equivalence classes for each such partition. The number of restricted SSRs can then be written as

$$N(\ell) = \sum_{k=1}^{4} C(\ell, k) \cdot o(k) \tag{3.2}$$

To derive the formula for $C(\ell, k)$, we first recall that the number of ways to partition n elements into k non-empty sets is known as the Stirling number of the second kind and is denoted by $\left\{ {n \atop k} \right\}$.[313] It can be computed using the formula

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n$$

Let j be the number of the $i(\ell)$ ℓ-mers that are assigned to S_0. Note this does not include the ℓ-mers that are self-complementary that are forced to be in S_0. Let $C(\ell, k, j)$ be the number of ways that $i(\ell)$ ℓ-mers can be partitioned into $k+1$ sets $S_0, \ldots, S_k$, such that j of the ℓ-mers go into $|S_0|$ and $S_1, \ldots, S_k$ to are non-empty. We need to consider several cases depending on the value of j:

- In the case that $j = 0$, we are partitioning the $i(\ell)$ inputs among non-empty sets $S_1, \ldots, S_k$. Then $C(\ell, k, j) = \left\{ {i(\ell) \atop k} \right\}$.
- In the case that $1 \leq j \leq i(\ell) - k$, there are $\binom{i(\ell)}{j}$ ways to choose which j ℓ-mers are in S_0, and $\left\{ {i(\ell)-j \atop k} \right\}$ ways to partition the remaining ℓ-mers into $S_1, \ldots, S_k$. Hence, $C(\ell, k, j) = \binom{i(\ell)}{j} \left\{ {i(\ell)-j \atop k} \right\}$.
- In the case that $j > i(\ell) - k$, it is impossible to partition the remaining k (or fewer) ℓ-mers into $S_1, \ldots, S_k$ such that the sets are non-empty. Recall that as we assume the dimension is k, each set must contain at least one element. Hence, $C(\ell, k, j) = 0$.

Putting this together into Equation (3.2), we get

$$N(\ell) = \sum_{k=1}^{4} o(k) \left(\left\{ {i(\ell) \atop k} \right\} + \sum_{j=1}^{i(\ell)-k} \binom{i(\ell)}{j} \left\{ {i(\ell)-j \atop k} \right\} \right)$$

For $\ell = 2$, we have $N(2) = 2,135$ restricted SSRs, which is several orders of magnitude smaller than the initial 5^{16} possible SSRs and allows us to test the performance of all of them. For order-3 SSRs we get $N(3) = 2.9 \cdot 10^{21}$ which much smaller than the full search space of $5^{4^3} \approx 5.4 \cdot 10^{44}$, for order-4 SSRs we get a similar reduction in search space with $N(4) = 9.4 \cdot 10^{84}$ as opposed to the full search space of $5^{4^4} \approx 8.6 \cdot 10^{178}$. For these higher order SSRs, although the restricted search space is much smaller than the full original one, it is still too large to exhaustively search.

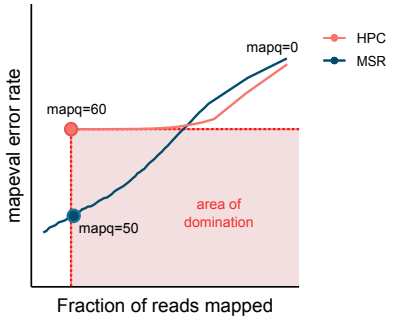


Figure 3.3.: **Illustration of how a respective mapq threshold is chosen for each of our evaluated MSRs.**
The orange dot shows the error rate and fraction of reads mapped for HPC at mapq threshold 60. Anything below and to the right of this point is strictly better than HPC 60, i.e. it has both a lower error rate and higher fraction of reads mapped. If an evaluated MSR does not pass through this region, then it is discarded from further consideration. In the figure, the blue MSR does pass through this region, indicating that it is better than HPC 60. We identify the leftmost point (marked as a blue dot) and use the mapq threshold at that point as the respective threshold.

3.2.3. Selection of mapping-friendly sequence reductions

We selected a set of "promising" SSRs starting from all of the SSRs enumerated in Section 3.2.2, that we call *mapping-friendly sequence reductions* (abbreviated *MSR*). The selection was performed on a 0.5x coverage read set, simulated from a whole human genome assembly.[4]The transformed reads were mapped to the transformed reference using `minimap2` and `paftools mapeval`[119] was used to compute a mapping error rate. Note that overfitting SSRs to a particular genome is acceptable in applications where a custom SSR can be used for each genome. Yet in this work, the same set of selected SSR will be used across all genomes.

For each evaluated SSR, we selected, if it exists, the highest mapq threshold for which the mapped read fraction is higher and the mapping error rate is lower than HPC at mapq 60 (0.93 and $2.1 \cdot 10^{-3}$ respectively), Figure 3.3 illustrates the idea. Then we identified the 20 SSRs that have the highest fraction of reads mapped at their respective thresholds. Similarly we identified the 20 SSRs with the lowest mapping error rate. Finally we select the 20 SSRs that have the highest percentage of thresholds "better" than HPC at mapq 60; i.e. the number of mapq thresholds for which the SSR has both a higher fraction of reads mapped and lower mapping error rate than HPC at a mapq threshold of 60, divided by the total number of thresholds (=60).

The union of these 3 sets of 20 SSRs resulted in a set of 58 "promising" MSRs. Furthermore, we will highlight three MSRs that are "best in their category", i.e.

- **MSR$_F$**: The MSR with the highest fraction of mapped reads at a mapq threshold of 0.
- **MSR$_E$**: The MSR with the lowest mapping error rate at its respective mapq threshold.
- **MSR$_P$**: The MSR with the highest percentage of mapq thresholds for which it is "better" than HPC at mapq 60.

Figure 3.4 shows the actual functions MSR$_F$, MSR$_E$, MSR$_P$. An intriguing property is that they output predominantly As and Ts, with MSR$_P$ assigning only 2 input pairs to the G/C output whereas MSR$_E$ and MSR$_F$ assign only one. Additionally, MSR$_E$ and MSR$_P$ both assign the {CC,GG} input pair to the deletion output ε removing any information corresponding to repetitions of either G or C from the reduced sequence. Overall this means the reduced sequences are much more AT-rich than their raw counterparts, but somehow information pertinent to mapping is retained

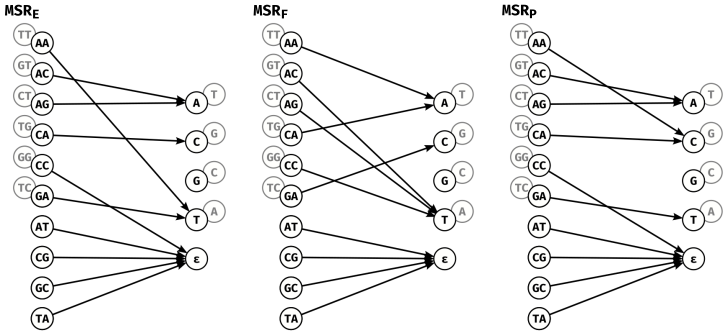


Figure 3.4.: **Graph representations of our highlighted MSRs: MSR$_E$, MSR$_F$, and MSR$_P$.**
MSR$_E$ has the lowest error rate of among MSRs at the highest mapq threshold for which it performs better than HPC at mapq 60, MSR$_F$ has the highest fraction of reads mapped at mapq 60 and MSR$_P$ has the highest percentage of mapq thresholds for which it outperforms HPC at mapq 60. The grayed out nodes represent the reverse complement of input dinucleotides and outputs, as in Figure 3.1C. For example for MSR$_E$, AA is mapped to T, so TT is mapped to A.

The 58 selected MSRs, as well as HPC and the identity transformation (denoted as *raw*), were then evaluated on larger read-sets simulated from 3 whole genome references: a whole human genome assembly,[4] a whole *Drosophila melanogaster* genome assembly[314] and a synthetic centromeric sequence[263] (see STAR Methods for more details).

3.2.4. Mapping-friendly sequence reductions lead to lower mapping errors on whole genomes

Across the entire human genome, at high mapping quality thresholds (above 50), our selected 58 MSRs generally have lower mapping error rate than HPC and raw Figure 3.5A and Table 3.1. This is not surprising, as we selected those MSRs partly on the criteria of outperforming HPC at mapq 60; however, it does demonstrate that we did not overfit to the simulated reads used to select the MSRs.

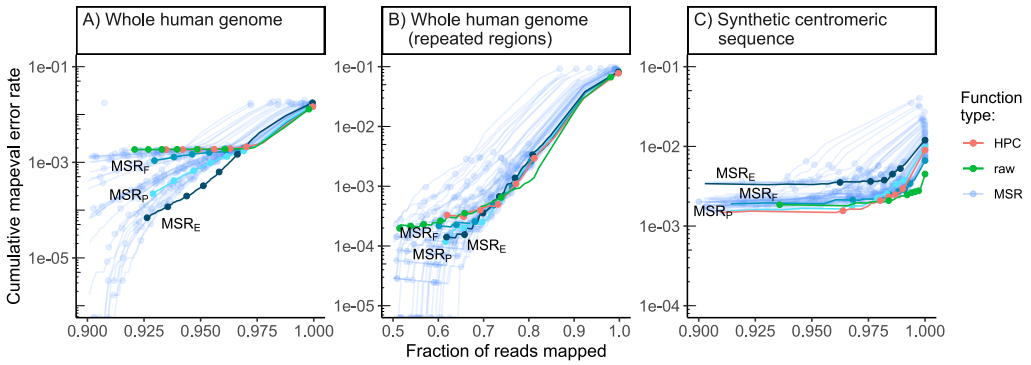


Figure 3.5.: **Performance of our 58 selected mapping-friendly sequence reductions across genomes on reads simulated by `nanosim`.**
Panel **A)** shows the whole human genome assembly, **B)** the subset of mapped reads from panel B that originate from repetitive regions, and **C)** the "TandemTools" synthetic centromeric reference sequence. We highlighted the best-performing mapping-friendly sequence reductions as MSR E, F and P, respectively in terms of cumulative `mapeval` error rate, fraction of reads mapped, and percentage of better thresholds than HPC. Each point on a line represents, from left to right, the mapping quality thresholds 60, 50, 40, 30, 20, 10 and 0. For the first point of each line, only reads of mapping quality 60 are considered, and the y value represents the rate of these reads that are not correctly mapped, the x value represents the fraction of simulated reads that are mapped at this threshold. The next point is computed for all reads of mapping quality ≥ 50, etc. The rightmost point on any curve represents the mapping error rate and the fraction of mapped reads for all primary alignments. The x-axes are clipped for lower mapped read fractions to better differentiate HPC, raw and MSRs E, F and P.

Mapping quality is only an indication from the aligner to estimate whether a read mapping is correct, and according to Figure 3.5A the mapping error rate of most MSRs is low even for mapping qualities lower than 60. Therefore, we choose to compare MSR-mapped reads with lower mapping qualities against raw or HPC-mapped reads with the highest (60) mapping quality (which is the mapping quality thresholds most practitioners would use by default).

Table 3.1 shows that the three selected MSRs outperform both HPC and raw in terms of mapping error rate, with similar fractions of mapped reads overall. For example on the human genome, at mapq≥ 50, MSR_F, MSR_P and MSR_E all map more reads than either HPC or raw at mapq=60, and MSR_P and MSR_E also have mapping error rates an order of magnitude lower than either HPC or raw.

To evaluate the robustness of MSRs E, F and P we investigated the impact of mapping to a different organism or using another mapper. To this effect we repeated the evaluation pipeline in these different settings:

- Using the *Drosophila melanogaster* whole genome assembly as reference and mapping with `minimap2`
- Using the whole human genome assembly as reference but mapping with `winnowmap2`(version 2.02).[120] The same options as `minimap2` were used, and k-mers were counted using `meryl`,[315] considering the top 0.02% as repetitive (as suggested by the `winnowmap2` usage guide).

MSRs E, F and P behave very similarly in both of these contexts compared to HPC/raw: by selecting mapped reads with mapq≥ 50 for the three MSRs we obtain a similar fraction of mapped reads with much lower error rates (Table 3.1). A noticeable exception is the `winnowmap2` experiment, where a larger fraction of raw reads are mapped than any other MSR and even HPC. A more detailed results table can be found in Table A.1, and a graph of MSR performance on the whole Drosophila genome in Figure A.6. As Figure A.6 shows, we also evaluated these MSRs on a whole *Escherichia coli* (Genbank ID U00096.2) genome, where we observed similar results, albeit the best MSRs do not seem to be one of our three candidates. This might mean that specific MSRs are more suited to particular types of genomes.

		Whole human genome `minimap2`		Whole human genome `winnowmap2`		Whole Drosophila genome `minimap2`	
	mapq	fraction	error	fraction	error	fraction	error
HPC	60	0.935 +0%	1.85e-03 + 0%	0.894 +0%	1.43e-03 + 0%	0.957 +0%	2.27e-03 + 0%
raw	60	0.921 −1%	1.86e-03 + 0%	**0.932 +4%**	1.75e-03 +23%	0.958 +0%	2.27e-03 − 0%
MSR_F	50	**0.938 +0%**	1.29e-03 −30%	0.886 −1%	3.82e-04 −73%	**0.960 +0%**	1.37e-03 − 39%
MSR_E	50	0.936 +0%	**1.17e-04 −94%**	0.820 −8%	**8.93e-05 −94%**	0.954 −0%	**0.00 −100%**
MSR_P	50	**0.938 +0%**	4.15e-04 −78%	0.845 −6%	1.14e-04 −92%	0.957 +0%	8.11e-04 − 64%

Table 3.1.: **Performance of MSRs, HPC, and raw mappings across different mappers and reference sequences.**
For each reference sequence and mapper pair, we report the fraction of reads mapped ("fraction" columns), the `paftools mapeval` mapping error rate ("error" columns). The percentage differences are computed w.r.t to the respective HPC value. For HPC and the raw these metrics were obtained for alignments of mapping quality of 60. For MSRs E, F and P these metrics were obtained for alignments of mapping quality ≥ 50.

3.2.5. Mapping-friendly sequence reductions increase mapping quality on repeated regions of the human genome

To evaluate the performance of our MSRs specifically on repeats, we extracted the reads for which the generating region overlapped with the repeated region of the whole human genome by more than 50% of the read length. We then evaluated the MSRs on these reads only. Repeated regions were obtained from `https://t2t.gi.ucsc.edu/chm13/hub/t2t-chm13-v1.1/rmsk/rmsk.bigBed`.

We obtained similar results as on the whole human genome, with MSRs E, F and P performing better than HPC at mapq 50 (Figure 3.5B). At a mapq threshold of 50, the mapping error rate is 53%, 31%, and 39% lower than HPC at mapq 60 for MSRs E, F and P respectively, while the fraction of mapped reads remains slightly higher. At mapq=60, raw has an mapping error rate 40% lower than HPC but the mapped fraction is also 17% lower.

3.2.6. Raw mapping improves upon HPC on centromeric regions

On the "TandemTools" centromeric reference, HPC consistently maps a smaller fraction of reads than raw, across all mapping quality thresholds (Figure 3.5C). Additionally, the mapping error rate for raw is often inferior to that of HPC. The same is true for our selected MSRs: most of them have comparable performance to HPC, but none of them outperform raw mapping (Figure 3.5C).

We conjecture this is due to the highly repetitive nature of centromeres. HPC likely removes small unique repetitions in the reads and the reference that might allow mappers to better match reads to a particular occurrence a centromeric pattern. Mapping raw reads on the other hand preserves all bases in the read and better differentiates repeats. Therefore it seems inadvisable to use HPC when mapping reads to highly repetitive regions of a genome, such as a centromere.

3.2.7. Positions of incorrectly mapped reads across the entire human genome

To study how MSRs E, F, and P improve over HPC and raw mapping in terms of mapping error rate on the human genome, we selected all the primary alignments that `paftools mapeval` reported as incorrectly mapped. For HPC and raw, only alignments of mapping quality equal to 60 were considered. To report a comparable fraction of aligned reads, we selected alignments of mapping quality ≥ 50 for MSRs. We then reported the origin of those incorrectly mapped reads on whole human genome reference, shown per-chromosome in Figure 3.6.

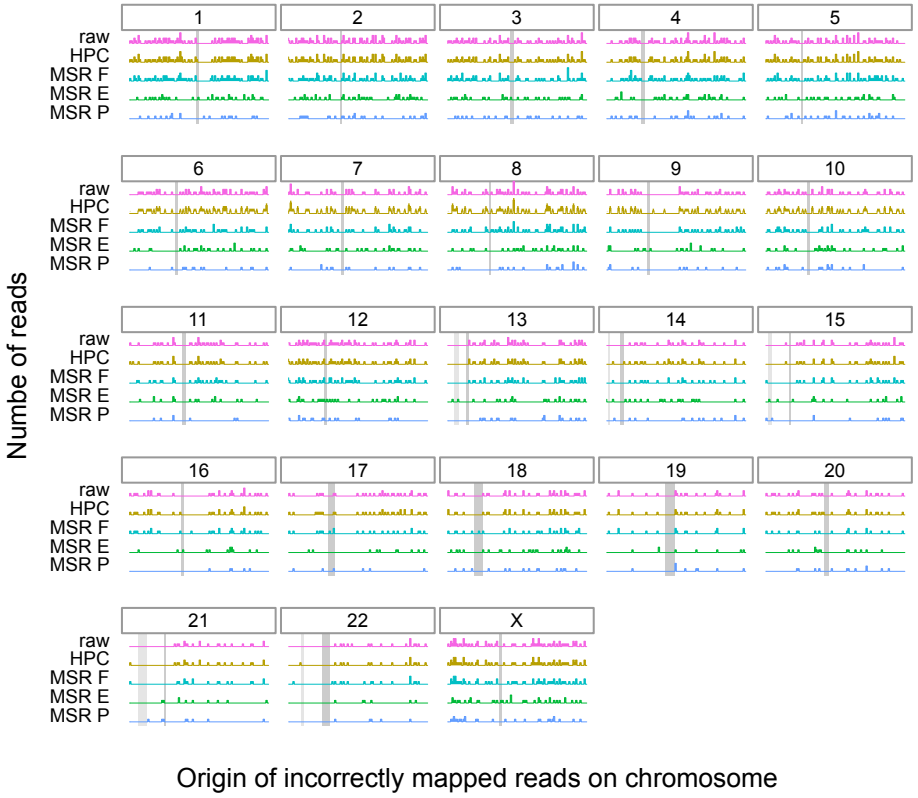


Figure 3.6.: **Histogram of the original simulated positions for the incorrectly mapped reads using `minimap2` at high mapping qualities across the whole human genome, for several transformation methods.**
For a given chromosome, each row represents the number of simulated reads starting at that particular region. The dark gray rectangle represents the position of the centromere for that chromosome, obtained from annotations provided by the T2T consortium (http://t2t.gi.ucsc.edu/chm13/hub/t2t-chm13-v1.1/). Similarly for chromosomes 13, 14, 15, 21 and 22, a lighter gray rectangle represents the position of the "stalk" satellites also containing repetitive regions. For HPC and raw reads only alignments of mapping quality 60 were considered. To provide a fair comparison, alignments with mapping qualities ≥ 50 were considered for MSRs E, F and P.

We observe that erroneously mapped reads are not only those from centromeres, and instead originate from many other genomic regions. MSRs E and P have a markedly lower number of these incorrect mappings than either HPC or raw, with 1118 incorrect mappings for raw mappings and 1130 for HPC as opposed to 549, 970 and 361 for MSRs E, F and P respectively. This stays true even for difficult regions of the genome such as chromosome X, where raw and HPC have 70 incorrect mappings as opposed MSRs E and P that have 39, and 27 errors respectively.

We also investigated where all simulated reads were mapped on the whole human genome assembly, for raw, HPC and MSRs E,F and P in Figures A.1 through A.5. The correctly mapped reads are, as expected, evenly distributed along each chromosome. The incorrectly mapped reads are however unevenly distributed. For most chromosomes there is a sharp peak in the distribution of incorrectly mapped reads, located at the position of the centromere. For the acrocentric chromosomes, there is a second peak corresponding to the "stalk" satellite region, with an enrichment of incorrectly mapped reads. This is expected since both centromeres and "stalks" are repetitive regions which are a challenge for mapping. For chromosomes 1, 9 and 16 however the majority of incorrectly mapped reads originate in repeated regions just after the centromere.

3.3. Discussion

We have introduced the concept of mapping-friendly sequence reduction and shown that it improves the accuracy of the popular mapping tool `minimap2` on simulated Oxford Nanopore long reads.

We focused on reads with high mapping quality (50-60), as it is a common practice to disregard reads with low mapping quality.[261,316,317] However across all mapped reads (mapq$\geq$ 0), HPC and our MSRs have lower mapping accuracies than raw reads, consistent with the recommendation made in `minimap2` to not apply HPC to ONT data. Despite this, we newly show the benefit of using HPC (and our MSRs) with `minimap2` on ONT data when focusing on high mapping quality reads. Furthermore MSRs provide a higher fraction of high-mapq reads compared to both raw and HPC, as shown on the human and Drosophila genomes.

A natural future direction is to also test whether our MSRs perform well on mapping Pacific Biosciences long reads. Furthermore, it is important to highlight that our sampling of MSRs is incomplete. This is of course due to only looking at functions having $l = 2$, but also to the operational definition of RC-core-insensitive functions, and finally to taking representatives of equivalence classes. An interesting future direction would consist in exploring other families of MSRs, especially those that would include HPC and/or close variations of it.

Additionally, our analyses suggests to not perform HPC on centromeres and other repeated regions, hinting at applying sequence transformations to only some parts of the genomes. We leave this direction for future work.

3.4. Limitations of this study

Our proposed MSRs improve upon HPC at mapq 60, both in terms of fraction of reads mapped and mapping error rate on whole human, *Drosophila melanogaster*, and *Escherichia coli* genomes. We chose these sequences because they were from organisms that we deemed different enough, however it would be interesting to verify if our proposed MSRs are still advantageous on even more organisms, e.g. more bacterial or viral genomes. This would allow us to assess the generalizability of our proposed MSRs.

We made the choice of using simulated data to be able to compute a mapping error rate. Some metrics, such as fraction of reads mapped might still be informative with regards to the mapping performance benefits of MSRs, even on real data. Evaluating the MSRs on real data might be more challenging but would offer insight into real-world usage of such pre-processing transformations.

The hypothesis we made in subsection 3.2.2.2 was derived from non mapping-related tests, it helped us reduce the search space and find MSRs. Testing if this hypothesis holds true on mapping tasks would help us make sure we are not missing some potentially well-performing SSRs by discarding them at this stage.

Finally, the restrictions we imposed to define RC-core-insensitive MSRs though intuitively understandable are somewhat arbitrary, so exploring a larger search space might be beneficial. Alternatively for higher order MSRs, even with our restrictions, the search spaces remain much too large to be explored exhaustively. To mitigate this problem, either further restrictions need to be found, or an alternative, optimization-based exploration method should be implemented.

Acknowledgements

The authors thank Kristoffer Sahlin for feedback on the manuscript.

R.C was supported by ANR Transipedia, SeqDigger, Inception and PRAIRIE grants (ANR-18-CE45-0020, ANR-19-CE45-0008, PIA/ANR16-CONV-0005, ANR-19-P3IA-0001). This method is part of projects that have received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreements No 956229 and 872539. L.B was also supported by the ANR PRAIRIE grant (ANR-19-P3IA-0001). This material is based upon work supported by the National Science Foundation under Grant No. 1453527 and 1931531.

Author contributions

Conceptualization, P.M. and R.C.; Methodology, L.B., P.M. and R.C.; Software, L.B.; Validation, L.B. and R.C.; Formal Analysis, L.B.; Investigation, L.B.; Resources, R.C.; Writing – Original Draft, L.B., P.M. and R.C.; Writing – Review & Editing, L.B., P.M. and R.C.; Visualization, L.B.; Supervision, R.C.; Project Administration, R.C.; Funding Acquisition, R.C.;

Declaration of interests

The authors declare no competing interests.

STAR Methods

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Rayan Chikhi (rayan.chikhi@pasteur.fr)

Materials availability

This study did not generate new unique reagents.

Data and code availability

This paper analyzes existing, publicly available data. These accession numbers for the datasets are listed in the key resources table

All original code has been deposited at a github backed zenodo repository and is publicly available as of the date of publication. DOIs are listed in the key resources table, and the backing github repository is available at github.com/lucblassel/MSR_discovery.

Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

Method details

3.4.0.1. Datasets

The following three reference sequences were used for evaluation:

1. **Whole human genome:** This reference sequence is a whole genome assembly of the CHM13hTERT human cell line by the Telomere-to-Telomere consortium.[4] We used the 1.1 assembly release (Genbank Assembly ID GCA_009914755.3).
2. **Whole *Drosophila* genome:** This reference sequence is a whole genome assembly of a *Drosophila melanogaster*, release 6.35 (Genbank Assembly ID GCA_000001215.4).[314]
3. **Synthetic centromeric sequence:** This sequence was obtained from the `TandemTools` mapper test data.[263] It is a simulated centromeric sequence that is inherently difficult to map reads to. Appendix A.1 describes how it was constructed, and it is downloadable from `https://github.com/lucblassel/TandemTools/blob/master/test_data/simulated_del.fasta`

3.4.0.2. Simulation pipeline

Given a reference sequence, simulated reads were obtained using `nanosim`[318] with the `human_NA12878_DNA_FAB49712_guppy_flipflop` pre-trained model, mimicking sequencing with an Oxford Nanopore instrument. The number of simulated reads was chosen to obtain a theoretical coverage of whole genomes around 1.5x, this resulted in simulating $\approx 6.6 \cdot 10^5$ reads for the whole human genome and $\approx 2.6 \cdot 10^4$ reads for the whole Drosophila genome. Since the centromeric sequence is very short, we aimed for a theoretical coverage of 100x which resulted in $\approx 1.3 \cdot 10^4$ simulated reads.

For each evaluated SSR, the reads as well as the reference sequence were reduced by applying the SSR to them. The reduced reads were then mapped to the reduced reference using `minimap2`[119] with the `map-ont` preset and the `-c` flag to generate precise alignments. Although HPC is an option in `minimap2` we do not use it and we evaluate HPC as any of the other SSRs by transforming the reference and reads prior to mapping. The starting coordinates of the reduced reads on the reduced reference were translated to reflect deletions incurred by the reduction process. The mapping results with translated coordinates were filtered to keep only the primary alignments. This process was done for each of our SSRs as well as with HPC and the original untransformed reads (denoted as *raw*).

3.4.0.3. Evaluation pipeline

We use two metrics to evaluate the quality of a mapping of a simulated read set. The first is the *fraction of reads mapped*, i.e. that have at least one alignment. The second is the *mapping error rate*, which is the fraction of mapped reads that have an incorrect location as determined by `paftools mapeval`.[119] This tool considers a read as correctly mapped if the intersection between its true interval of origin, and the interval where it has been mapped to, is at least 10% of the union of both intervals.

Furthermore, we measure the mapping error rate as a function of a given *mapping quality threshold*. Mapping quality (abbreviated mapq) is a metric reported by the aligner that indicates its confidence in read placement; the highest value (60) indicates that the mapping location is likely correct and unique with high probability, and a low value (e.g. 0) indicates that the read has multiple equally likely candidate mappings and that the reported location cannot be trusted. The mapping error rate at a mapq threshold t is then defined as the mapping error rate of reads whose mapping quality is t or above. For example, the mapping error rate at $t = 0$ is the mapping error rate of the whole read set, while the mapping error rate at $t = 60$ is the mapping error rate of only the most confident read mappings. Observe that the mapping error rate decreases as t increases.

All experiments performed for this article are implemented and documented as nextflow workflows available in this project's repository (`https://github.com/lucblassel/MSR_discovery`). These workflows may be used to rerun experiments and reproduce results. The repository also contains a Rmarkdown notebook to generate all figures and tables in the main text and supplemental information from the pipeline outputs.

Supplementary information

Supporting Information can be found in Appendix A

4. Learning From Sequences and Alignments

Sequences and sequence alignments are very rich sources of information. As was stated in Chapters 2 and 3, many downstream analyses rely on sequence alignments.

In whole genome assembly, where sequencing reads are combined together to deduce the genome sequence, pairwise sequence alignment is used, both in reference-based[319,320] and *de novo*[321,322] assembly. It has also been used to deduce protein function.[323] Pairwise alignment, has been used for sequence clustering[116] as well as detecting genetic[324] and structural variants.[325,326] Multiple sequence alignments are also very widely used, mainly in phylogenetic analyses where the evolutionary history of a set of sequences are studied and represented as trees,[327,328] but they have also been used extensively in protein structure prediction.[329]

More recently, as computational power and datasets have grown, more and more machine learning methods are being used on sequence alignments in order to gain biological insight. In this chapter, we will explore how this can be done, as an introduction to Chapter 6 where we present an application: predicting HIV drug resistance mutations.

4.1. What to learn ?

One of the first questions one might ask themselves when wishing to use machine learning with sequence data is "what can I learn?". A simplistic answer to this question would be "a lot of things" as the following section will strive to show. To choose what we learn we must first choose a learning paradigm.

4.1.1. Supervised learning from biological sequences

Supervised learning is one of the main machine learning paradigms, where we have data that consists of a collection of input and output pairs (e.g. a DNA sequence and an associated species). By feeding these pairs to our algorithm of choice, it will learn to predict the output based on the input alone. This is a very powerful way of learning something interesting. We can consider the link between inputs and

outputs as extra knowledge that the dataset creator or curator can infuse in the learning algorithm. Within the supervised learning paradigm there are two possible tasks: *regression* and *classification*.

4.1.1.1. Regression tasks

For regression tasks, the outputs of our input-output pairs are encoded by a continuous numerical value. Regression models will therefore output continuous real values. Fortunately, many interesting continuous values can be computed from aligned sequences, and in many cases, machine learning models can be trained to predict these variables.

Regression methods have been used to predict drug response in cancer patients[330] and resistance levels to drugs in HIV.[331] These methods are also extensively used in protein structure prediction where they are trained to predict residue angles or values in protein contact maps from aligned sequences,[332–336] or directly from an MSA.[137] Regression algorithms have been used to predict protein fitness *in silico*[337–339] to speed up protein engineering, and make some processes like drug development faster and cheaper. They have also been used in many other tasks such as predicting gene expression levels[340] or predicting multiple sequence alignment scores.[341]

In many cases these methods use an encoded representation of the sequences (c.f. Section 4.3) as input, but some represent the inputs as values computed from alignments. For example, protein structure can be predicted from contact maps[342] derived from MSAs, and gene expression levels can be predicted from lists of mutations that are obtained through alignment to a reference sequence.[340] This last approach is also used in Chapter 6 to predict drug resistance in HIV.

4.1.1.2. Classification tasks

For classification tasks, the outputs of our input-output pairs are categorical in nature and often represented as discrete integer values. Originally, most classification methods were designed for binary classification with only two possible outputs: a "positive" and a "negative" class. This is a simpler problem to solve than multiclass classification problems where more than two outputs are possible. Most methods that can handle binary classification have been adapted to multiclass classification.

In biology, categorizing and classifying is often at the root of several research problems. As such, machine learning classifiers have obvious applications and have been widely used with sequence data as inputs. Classifiers have been used to predict if a particular virus[343,344] (also Chapter 6), or bacteria[345,346] is resistant to antiviral or antimicrobial drugs respectively. Some classifier models have also been used to predict characteristics at positions in a sequence, like methylation site prediction[347] or splicing site detection.[348] This type of approach has also been applied for sequence

labeling tasks, where each token of a sequence *(amino acids, codons, …)*, is assigned a label using classifiers. This has been applied to protein secondary structure prediction[349] where each residue is assigned a label (α-helix or β-sheet), or gene prediction where stretches of the sequence are classified as part of a gene or not.[350,351] Base calling for Nanopore sequencing data (mentioned in 1.2.3), is also a sequence labeling task, although the sequence is made up of voltages and the assigned labels are nucleotides. Finally, classifiers have also been used to predict more general characteristics of a given sequence, like the cellular localization[352] and putative function[353] of proteins, or the cellular localization of gene expression data.[354]

I have presented here, only a fraction of what is possible to learn from sequences in the supervised learning paradigm. I hope you will agree with me that there is no shortage of problems in computational biology that are suited to this sort of approach. By using machine learning, instead of more formal statistical approaches, there is a lower amount of upfront assumptions and the algorithm is tasked with figuring out what features of the data are important or not for the task at hand.

4.1.2. Unsupervised learning from biological sequences

The second main machine learning paradigm is called, by contrast to supervised learning, unsupervised learning. In this paradigm we do not have input-output pairs but only inputs. The goal of unsupervised machine learning methods is to extract some structure or patterns from the given input without additional guidance.

One of the main tasks in the unsupervised learning paradigm is clustering, wherein similar inputs are grouped together, methods like k-means or hierarchical clustering[355] often use some type of distance metric between inputs to define clusters of similar inputs. Clustering can be used for classification tasks, indeed if some characteristics of sequences in a given cluster are known then we can make the assumptions that sequences in the same cluster will be similar and share these characteristics. This has been used to group proteins in families.[356] Clustering methods can also be used to remove duplicate or near-duplicate sequences in datasets.[357] Phylogenetic trees can be considered as a specific type of clustering methods, and they have been used to group biological sequences.[358]

One of the main obstacles to clustering biological sequences is the need for computing distances between sequences. As stated in Chapter 2, obtaining a biologically relevant distance metric between two sequences, such as the edit-distance, is no easy task. Additionally, in many cases, all pairwise distances are needed for clustering, meaning at least a quadratic time and space complexity for a naive clustering algorithm. Two approaches can be used to resolve this problem: devise methods that do not need all pairwise distances,[359] or find a way to speed up distance computation. Some methods have been developed to devise distance metrics that are biologically

relevant and less expensive to compute than the edit-distance: like the hashing based MASH[360] or dashing,[361] or the neural network based NeuroSEED.[362]

Unsupervised learning can also be used without clustering. For example, unsupervised methods based on maximum likelihood approaches have been used to predict mutational effects in protein sequence[363] as well as predict recombination hotspots in human genomic sequences.[364]

In many cases, unsupervised learning can be done as a preliminary dimensionality reduction step to a supervised learning task. Indeed biological data is often high-dimensional, and it is often useful to lower the amount of dimensions to speed up computations. Some unsupervised methods can reduce the number of dimensions while retaining most of the information. One such method, Principal Component Analysis (PCA), is widely used. PCA has been applied to distance matrices to compute phylogenetic trees,[365] and work has been done to apply PCA directly to MSAs without needing to go through a distance matrix.[366] PCA is also widely used in clustering applications.[367–370]

4.1.3. Others paradigms

More recently, other learning paradigms have gained popularity in machine learning circles. Within the *semi-supervised* paradigm, a small amount of labelled data (*i.e.* input-output pairs) is included in a large un-labelled dataset, and methods can leverage both. This approach has been used to predict drug to protein interactions[371] and predict the secondary structure of specific transmembrane proteins.[372]

In the *self-supervised* paradigm, models are first trained on a proxy task that hopefully makes use of important information in the data. Through this pre-training step, self-supervised models extract important information from the data and create internal features and models that can then be leveraged in a supervised or unsupervised fine-tuning task. This paradigm has exploded lately within the field of natural language processing and machine translation with the rise of transformers, but has also been widely used to create protein language models like ProtBert[373] and extract information from disordered protein regions.[374] We will look at self-supervised learning in a little more detail in Chapter 7.

Finally, the end-to-end learning paradigm designates the process of chaining several machine learning tasks together and optimizing the algorithms simultaneously using the error from the loss of the last task of the group. This has been successfully used to predict protein-protein interaction surfaces in three dimensions[375] as well as predict micro-RNA targets sequences.[376] This paradigm can also be used in a task-based fashion, where a differentiable loss function is crafted on a traditionally non-machine learning task and used to train preceding models. This has been explored for sequence alignment and is further detailed in Chapter 7

4.2. How to learn ?

Machine learning regroups a multitude of techniques and methods to extract knowledge and make data-driven predictions. In this section, we will quickly go over some of the main supervised-learning methods, and go into more detail for techniques used in Chapter 6: logistic regression, naive Bayes and random forests.

4.2.1. General setting

Supervised machine learning is an optimization process. A given algorithm, which I will refer to as a *model*, has an associated loss function that can be evaluated on a dataset. This loss represents how well the model is predicting outputs from inputs on known input-output pairs. Through an iterative process, this loss is optimized *(in our case minimized)* over all pairs forming a dataset. Often, in the literature, loss and cost are used interchangeably.[377] I will favor loss in the following sections.

There is no shortage of loss functions,[378] some of them are specifically crafted for a given model while some are widely used in regression tasks like the Root Mean Square Error (RMSE). Others like the cross-entropy loss are used in classification tasks.

After training a machine learning model on a dataset, it is often important to compute a performance measure to get an idea of how well this model is performing. We could do this on the same data on which the model was trained, this would however be wrong. Indeed, it gives an unfair advantage to the model since it predicts outputs from examples it has already seen. Furthermore, it gives us no insight into the generalizability of the model since it could just learn the dataset by heart, getting a perfect score on it while being completely useless on new unseen data. This situation is known as *overfitting*,[355] shown in Figure 4.1. Since being able to predict outcomes on unseen data is the main goal of a machine learning model, we need another way of measuring model performance. The way machine learning practitioners can measure the performance of their model in a more unbiased manner is by separating the dataset into two parts before even starting to train the model: one part (usually the majority of the data) is used as the *training set*, and the other as the *testing set*. Logically, the training set is used to train the model while the testing set is used to evaluate the performance of the model after training.

As there is a multitude of loss functions, there are many performance metrics to assess how the model is doing on the testing data, especially for classification tasks.[379] For regression, RMSE is also widely used as a performance metric, along with the Mean Absolute Error (MAE). For classification, accuracy is the most widely used performance metric. Accuracy is the ratio of the number of correctly classified examples divided by the total number of examples. Accuracy has also been adapted to specific settings like unbalanced data where the different possible output classes are

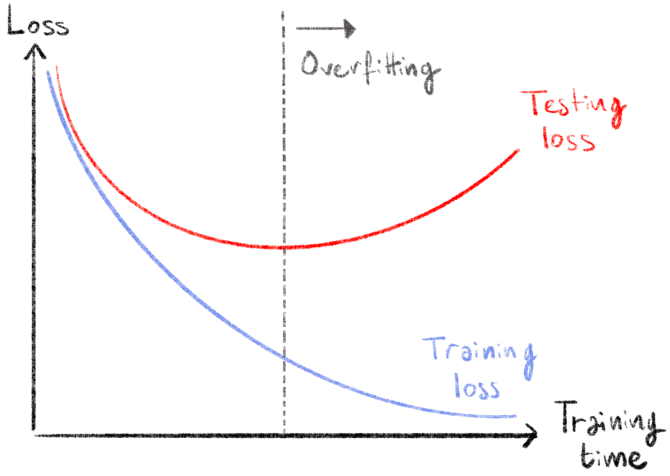


Figure 4.1.: **Overfitting behaviour in loss functions.**
The two curves show how the loss calculated on the training set (blue) and the testing set (red) evolve as training time increases. At first both decrease showing that the model learns informative and generalizable features. At some point, training loss keeps decreasing and testing loss increases, meaning that the model is learning over-specific features on the training set and is no longer generalizable: it is overfitting.

not represented equally.[380] The testing set must stay completely separate from the training set and decisions about model settings or input features used must be made without help of the testing data. If these stringent conditions are not respected this can lead to *data leakage* and artificially increase performance of the model on the testing data, giving us a biased view of the model's performance and generalizability.[381] This leaking of testing data into the training process is a common pitfall of machine learning.[382] To remedy to this problem, a separate dataset is often reserved and used as a *validation set,* in order to provide some estimation of model performance without using the testing set.

In many cases, machine learning models have a number of parameters that guide model behavior. These parameters are chosen before training and are different from the internal parameters of the model that are optimized during training. As such, they are often called *hyper-parameters*. These could, for example, be the number of levels in a decision tree, a learning rate, or a stopping threshold. The value of these hyper-parameters is often very influential on model performance. However, setting hyper-parameter values based on the model's test set performance would lead to data leakage as stated earlier, and using a separate validation set can lead to small training sets. To still be able to tune hyper-parameters for optimal performance, and keep a large training set, *k-fold cross-validation* is used.[355] In this setting, shown in Figure 4.2, the testing set is set aside before model training and reserved for the

final model performance evaluation. The training set is then further subdivided into k equally-sized subsets, called folds. Each of the k folds is then used to create a what is called a *validation split*: the fold acting as a within-split testing set and the rest of the general training set is used as the within-split training set. This results in k pairs of disjoint training and testing sets, and each example of the general training data is used exactly once in a within-split testing set. An idea of the model performance can be obtained by measuring performance on the within-split testing sets and averaging the measures. This cross-validation performance can be used to inform hyper-parameter value choice without using the reserved testing set and avoiding data leakage.

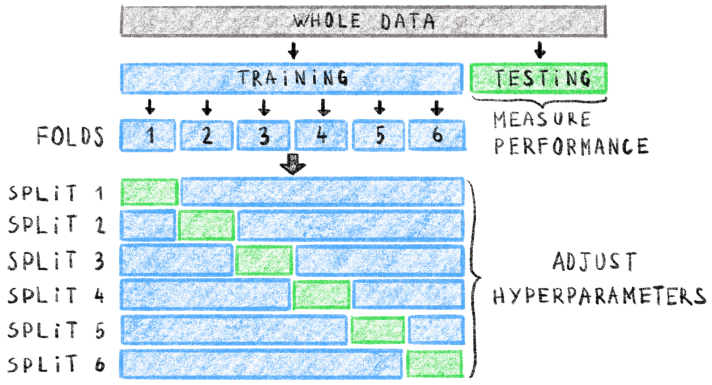


Figure 4.2.: **Example of data splits into training, testing and validation sets with 6-fold cross-validation.**
In this setting, the whole data set is first split into a training and testing set. The testing set is kept separate to assess final model performance. The training set is split into 6 folds resulting in 6 splits. In each split of the training set, the correspoding fold is used as the within-split test set (green), and the rest of the training set is used as the within-split training set (blue). You can get an idea of the model performance by averaging measures on within-split testing sets and adjusting hyper-parameters accordingly, without using the global, reserved testing set. Adapted from https://scikit-learn.org/stable/modules/cross_validation.html

This is the general setting in which a lot of the supervised learning approaches in computational biology reside, *e.g.* cross-validation was used to tune hyper-parameters for the models in Chapter 6.

4.2.2. Tests and statistical learning

Some of the simplest models possible are derived from statistics and based on probabilities. One such way to classify data is with a statistical test, like Fisher's exact test[383] or a χ^2 test,[384] depending on the number of training examples. If one of

the input variables is significantly related to the output then one can make a crude prediction on the output based solely on the value of one input variable. By testing several features and predicting the output from a set of significantly related input variables (e.g. through a vote), then prediction accuracy can be improved. This approach is used as a baseline in the study presented in Chapter 6. It is, however, not very sophisticated and does not have the best predictive power.

A model that fits more squarely in the process of supervised learning described above is linear regression. This regression model assumes that the output value results from a linear combination of the input features and an intercept value. The coefficients of this linear combination and the intercept are the parameters that the models optimizes during the learning process. Often, the loss function used to fit this model is the RMSE mentioned above. The gradient of the RMSE w.r.t. all the coefficients of the model is easily derived and can be used for optimization. Since this model is very simple there is an exact analytical solution to find the minimum gradient value.[355] However, in some cases a gradient descent approach can be beneficial to train this model. This model has also been adapted to binary classification, by considering that the output value results from a linear combination of input models, passed through a logistic function. The resulting model is called logistic regression, and is one of the classifiers used in Chapter 6. Equations (4.1) and (4.2) show the mathematical model of linear and logistic regression respectively. In these equations, $\hat{y}^{(i)}$ represents the predicted output of the i^{th} example and $x_j^{(i)}$ the j^{th} variable of the i^{th} example input. θ_0 is the intercept and θ_j the coefficient corresponding to the j^{th} input variable.

$$\hat{y}^{(i)} = \theta_0 + \sum_{j=1}^{k} \theta_j \cdot x_j^{(i)} \tag{4.1}$$

$$\hat{y}^{(i)} = \frac{1}{1 + e^{-(\theta_0 + \sum_{j=1}^{k} \theta_j \cdot x_j^{(i)})}} \tag{4.2}$$

The model in Equation (4.1) outputs a continuous value used in regression, and the model in Equation (4.2) outputs a continuous value bounded between 0 and 1, that we can consider a probability of being in one of the classes. With this probability it is easy to classify a given example in one of the two classes. It is easy to extend the logistic regression model to multiclass classification, by training several models and predicting the class with the maximal probability.

These linear models are simple, but can achieve good performance. They can, however, be prone to overfitting. This often translates into very large values for the θ coefficients. In order to counter this, regularized versions of linear and logistic regression were introduced by adding the weights to the loss function in some way. By adding the coefficient values to the loss they are kept small through the optimization

process, reducing the risk of overfitting. The two main regularization strategies are the ridge[385] and Lasso[386] penalties.

The final supervised model I will present in this section is the Naive Bayes classifier. As its name indicates, it is based on Bayes' theorem of conditional probabilities. By making a strong assumption, that all variables of the input examples are mutually independent, we can derive the probability of the i$^{\text{th}}$ input example belonging to class C_α as:

$$p(C_\alpha|x_1^{(i)}, \dots, x_k^{(i)}) = Z \cdot p(C_\alpha) \prod_{j=1}^{k} p(x_j^{(i)}|C_\alpha) \tag{4.3}$$

With Z a constant that can be computed from the training data. Therefore it is very easy to use this to build a classifier by computing the probabilities of an example belonging to a class for all possible classes in the training data and assign the one with the maximal probability. In practice this is a very flexible model, since any probability distribution can be used for each feature and class. The parameters of these distributions can be learned with a maximum likelihood approach for example. This model builds upon the naive assumption (hence the name) that all input variables are mutually independent. This assumption is very often violated, especially in biological sequence data where independence is not at all guaranteed by the evolutionary process. This model is, however, quite robust to this, and stays performant despite the violations of this assumption.[387,388]

4.2.3. More complex methods

While these simple methods are quite useful in many settings, more complex methods were developed. One of the most popular methods, up until fairly recently, were Support Vector Machines (SVM). This classifier was first developed in 1982[389] and it functions by finding the optimal separation hyperplane between 2 classes, i.e. a linear boundary in high-dimensional space between training examples of two classes. What made it so popular is when it was associated with the so-called *kernel trick*.[390,391] With the kernel trick, training examples that cannot be linearly separated can be cheaply projected into a higher dimensional space, where linear separation is possible. This made SVMs very powerful and popular, and it was quickly adapted for regression tasks as well.[392] The main model that will interest us in this section however is not the SVM.

Random forests are another very popular model used for both classification and regression. As it is used in Chapter 6, we will go over it in more detail. Developed in the early 2000's,[393] it builds upon previous work: Classification And Regression Trees (CART).[394] CART decision trees are very useful for supervised learning tasks. To use CART trees, start at the root and at each node there is a condition on a

single input feature. This condition decides if the considered example goes to the right child or the left child. By traversing this tree, choosing the path through the conditions at all the nodes met by the example, we can assign the example to one of the leaves, corresponding to a class or a predicted value. An example of such a tree is given in Figure 4.3.

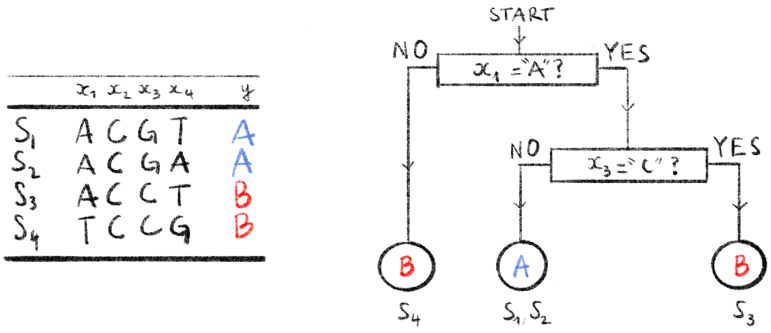


Figure 4.3.: **Example of a decision tree for DNA data.**
Here we have a dataset of 4 DNA sequences $S_1, \ldots, S_4$. Each sequence has 4 input variables $x_1, \ldots, x_4$, and an output variable y indicating the strain of a sequence. Each sequence can be classified by the decision tree on the right by following a path, from root to leaf, according to the conditions in internal nodes. The predicted strains are shown in the leaves. Sequences that end up in a given leaf node are indicated underneath that leaf node. This tree can classify these 4 sequences perfectly.

It is actually quite simple to build CART trees, the whole methods lies upon the principle of minimizing impurity *(or maximizing purity)* on a given input variable in child nodes. Impurity can be defined in many ways:[355] for regression it is often the Residual Sum of Squares (RSS), for classification it is often the Gini index or an entropy measure. Regardless of the chosen metric, a high impurity denotes a heterogeneous collection of examples and a low impurity indicates a homogeneous set of examples. When building the tree, recursively from the root, we find the condition, and the input variable on which the condition relies, by looking at all possible splits and choosing the one that decreases impurity the most in the child nodes. This process is continued recursively until the leaves are completely pure (likely resulting in overfitting) or until a certain stopping condition is met *(e.g.* purity threshold, maximum depth, *…)*. To avoid overfitting, trees can also be pruned after the building phase. CART trees have the distinct advantage of being interpretable: it is easy to figure out *why* an input has been assigned to a certain class, which can be very useful in biology or medicine.[395]

Despite these good properties, it is easy to overfit with decision trees, and small changes to the training data can induce large changes in the resulting tree,[355,395]

hurting interpretability. This is why the Random Forest (RF) model was created. RFs are essentially an ensemble of decision trees, *a forest*, built from the training data. To build one of the decision trees in a random forest, the training data is first bootstrapped: a new training set is sampled with replacement from the original training data with the same number of examples. This process is called *bagging* for "bootstrap aggregating". With this procedure, each decision tree is built from slightly different training data and will therefore likely be slightly different. An additional step to ensure some variability between the trees is in the choice of the splitting condition at each tree node. Where, in CART trees, all input variables are considered to find the optimal split, in RF trees, only a random subset of the input variables are considered at each node. This results in a set of decision trees that are all trained from slightly different data, with slightly different features at each node but that all have the same task on the training data. We can get a prediction from all these trees, by taking the majority predicted class for classification trees, or the average of predictions for regression trees.

All these measures to reduce the variance linked to decision trees, and to yield more generalizable models, make random forests very popular. They are often very competitive and often have better performance than the models presented above.[396,397] Furthermore, by only considering a subset of features at each tree node, RF often deals better with high-dimensional data than other methods.[397] Further refinements to the algorithm such as boosting, where misclassified examples are more likely to be selected in the bagging training sets have been very useful as well.

Deep learning has been used more frequently and more broadly to get good results across a large number of tasks. This is also true in biological contexts. However, Chapter 6 does not make use of deep learning methods so they will no be discussed here. A short introduction to deep learning will be presented in Chapter 7.

4.3. Pre-processing the alignment for machine learning

By now you will surely have noticed that all the models I presented above (with the exception of RFs) need to be trained on a collection of numerical variables, *i.e.* numerical vectors. Biological sequences, however, are not vectors of numbers. We therefore need to transform our sequences of letters into numerical sequences that we can feed to the machine learning model in this digestible form. Most supervised machine learning algorithms expect a matrix as input, where the rows are individual training examples and the columns numerical variables. A vector where each entry corresponds to an expected output value is used during training. In this section, I will present a few encoding methods, that transform a multiple sequence alignment in a matrix. Most of the encoding methods are not defined on an alignment, but on sequences alone. However, to represent these sequences they often need to have the same length, and for models to learn anything meaningful the values in features

should encode the same information across sequences. Therefore, prior to the encoding methods described below the sequences often need to be aligned to each other so that a specific position in a sequence is homologous to that position in all other training sequences.

4.3.1. General purpose encodings

The letters making up biological sequences are a form of categorical data. This type of variable is not specific to biology and as such, there exists many encoding schemes[398] to transform categorical variables into numerical vectors.

The most basic, and conceptually simple way to do so is to use the labeling scheme, often called ordinal encoding. Each level of the categorical variable is assigned an integer label. For example, when dealing with DNA sequences, we could have A=1, C=2, G=3 and T=4. This scheme outputs vectors that have the same size as the input sequence and going from the sequence to the encoded vector (and *vice versa*) is very easy. This encoding scheme has been used to predict resistance levels of HIV to antiviral drugs from sequencing data.[331] There is, however, a major disadvantage with using this method. As its name indicates, ordinal encoding implies that there is an order to the categorical variable levels (*e.g.* T>A) which, by definition, does not exist.[399–401] Another option is to use what I will refer to as binary labeling, where the categorical levels are first assigned an integer label which is then converted to a binary vector. If we use the ordinal DNA encoding from above and convert it to binary vectors we would get: A=$[0, 0]$, C=$[0, 1]$, G=$[1, 0]$ and T=$[1, 1]$. This type of representation is frequently used to represent gapless sequences, like k-mers, in a compressed form[402,403] (a character now only needs 2 bits instead of a full byte). For amino acids, since there are more characters, this encoding yields vectors of 5 bits.[404] Fundamentally, this encoding scheme has the same problem as the ordinal encoding, creating an order that does not exist, although with the order being split into separate values it can mitigate this effect a little bit.

One of the most widely used categorical encoding schemes, One-Hot encoding (OHE) (sometimes called orthonormal encoding[405]), does not have this ordering issue. The way OHE works is by creating a sparse binary vector of length d to represent a variable with d levels *(for DNA* $d = 4$*)*. If the i^{th} level of the categorical variable is to be encoded, then the i^{th} position in the vector is set to 1 and the rest set to 0. For example, if we consider that A is the first level of our variable then OHE would yield the following vector: $[1, 0, 0, 0]$. This encoding scheme has been used from the 1980's[406] to now,[407] and is the scheme used in Chapter 6. The performance of OHE can be on par with ordinal encoding,[408] but it is easily interpretable, which is often very important in biology since there is a one to one correspondence between a categorical value and a numerical feature. The main problem with OHE is that it tends to increase the number of features quite a lot, since the encoded vector

representation of a length n sequence is of length $n \times d$. An example comparing Ordinal, Binary and One-Hot encodings can be seen in Figure 4.4.

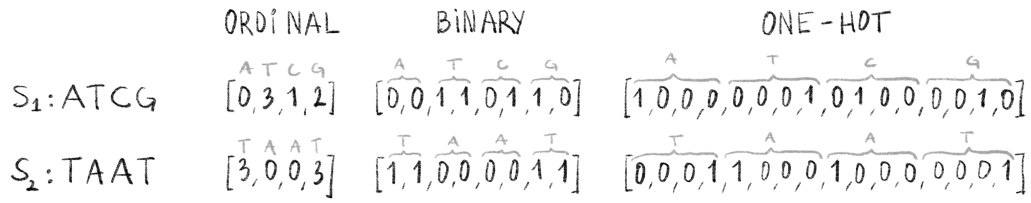


Figure 4.4.: **Example of 3 general categorical encoding schemes.**
Two sequences, `ATCG` and `TAAT` are shown encoded in three different encoding schemes: ordinal, binary and one-hot encoding. In the ordinal encoding, each character is assigned an integer value, here A=0, C=1, G=3 and T=4. In the binary encoding, these integer values are encoded with 2 bits. In the one-hot encoding scheme, a character corresponds to a sparse vector indicating which level of the variable is present: here A=[1,0,0,0]. Ordinal encoding preserves the dimension of the sequence while binary and one-hot encoding result in vectors with a bigger dimension than the original sequence.

These three general purpose encodings are but some of many,[398] and since categorical variables are often used in machine learning applications, these encodings are often available in widely used software libraries.[409]

4.3.2. Biological sequence-specific encodings

While the general-purpose encoding schemes presented above work well enough in practice, some specific encoding methods were developed to include some biological information in the sequence encodings that hopefully machine learning models will be able to leverage during training. These encodings have mostly been developed for encoding protein sequences, using physicochemical properties of amino acids.[404]

AAIndex[410] is a public database containing amino acid indices, *i.e.* sets of 20 numerical values (one for each AA) measuring some physicochemical property. There is a wide range of these indices, from hydrophobicity to flexibility or residue volume measures. By selecting an informative subset of 9 of these measures,[411] an amino acid can be represented by a length 9 numerical vector. In some cases, amino acids can be represented by all the 566 properties of AAIndex, and through PCA the dimension of the resulting numerical vectors can be reduced.[412] This biological sequence specific encoding has been implemented in a software library for biological sequence encoding.[413]

Another biological sequence-specific encoding is based on the Amino Acid classification Venn diagram defined by Taylor in 1986,[414] which groups amino acids into

eight different groups based on physicochemical properties: aliphatic, aromatic, hydrophobic, polar, charged, positive, small and tiny. With this classification, a single amino acid can be represented by a vector of length 8, each element representing a group, set to one when the amino acid belongs to the group and to zero when it does not. This encoding method was used as early as 1987 to predict secondary structures of proteins.[415] Later on, another five groups were proposed and used to encode each amino acid with longer vectors.[416]

A third encoding method, named BLOMAP,[417] encodes sequences based on values from the BLOSUM62 substitution matrix presented in Section 2.1.3. BLOMAP is defined by using a non-linear projection algorithm to generate vectors of length five, that capture the similarity measures contained in BLOSUM62. This encoding has been used to successfully predict cleavage sites of the HIV-1 protease[405,417] *(c.f. Section* 5.3.2.2). Other encodings such as OETMAP[418] have been derived from BLOMAP.

These three encodings are far from being the only ones specific to biological sequence. Many other encoding schemes were developed to learn from this type of sequence data. Some schemes do not encode positional data, and as such, can be applied to unaligned sequences. The simplest of these would be to represent a sequence by its amino acid, or k-mer frequencies. The latter is often referred to as n-gram encoding[419] and widely used, although with very short k-mers since the dimension of the encoding grows exponentially with k. With 20 amino acids, this encoding results in vectors that have a length of 20^k. Other encoding schemes use codon information to encode amino acids. One such scheme was proposed in,[404] where an amino acid is represented by a directed graph where vertices are nucleotides and edges represent paths needed to represent codons that code for that amino acid. This graph can then be converted to a 16-dimensional vector by flattening the corresponding adjacency matrix and used as an encoding method.

During the work that led to Chapter 6, several encoding methods were tested: Ordinal, Binary, OHE, AAIndex and Group encodings. The same two training sets of sequences were encoded using each of these methods, and 10 RF models were trained on each of the encoded datasets, om a binary classification task. Accuracy, precision and recall metrics were used to evaluate the performance of the RF on each encoded dataset. According to these metrics, the RF model had the best performance on the datasets encoded with OHE. OHE, also has the advantage of being more easily interpretable. As such, it was chosen for the work presented in Chapter 6.

Other encodings have been used to convert a biological sequence into a single real value. An encoding method based on chaos game theory[420] allows for a bijective mapping between the DNA sequence set and the real numbers set. This encoding is not specific to alignments and can be used to do alignment-free comparisons, as such it has been used often in bioinformatics applications.[421] Recently, this encoding scheme has been used to classify SARS-CoV2 sequences,[422] predict anti-microbial resistance from sequence data[345] and for phylogenetic analysis.[423]

In recent years algorithmic developments, computing power increase and the massive amounts of available data have made deep learning methods useful, possible to train and very popular. This has given rise to new sequence encoding methods, that are learned on the training data. These are often referred to as embeddings rather than encodings. Since these learned embeddings are not used in Chapter 6, for the sake of thematic coherence I will not be mentioning them here. I will, however, go over these embedding methods shortly in Chapter 7.

4.4. Conclusion

Alignments, and the sequences within them, are rich sources of information, that have long been exploited widely for many different types of analyses. With the rise of machine learning in the last years, it is logical that machine learning models have been applied more and more frequently to biological sequence data. Machine Learning is a wide field with many different methods and paradigms. Even simple methods like linear regression or naive Bayes can be very useful, and more complex models like random forests have been able to make very good predictions on biological data. The model is not the only variable to take into account when looking to apply machine learning methods on sequence data. Different encoding methods will yield different vector representations, with different characteristics and applications. Special care must therefore be given to the choice of biological sequence encoding scheme, prior to starting a machine learning analysis.

5. Viruses, HIV and Drug Resistance

5.1. What are viruses ?

Viruses occupy a strange place in the tree of life, with many debating if they are actually alive or not. André Lwoff gave what is probably the most fitting definition: *"viruses are viruses"*.[424] Despite this ambiguity, viruses share some common characteristics which allow us to define them as intracellular parasites:[425]

1. Viruses have some type of genetic information, contained in DNA or RNA.
2. This genetic information is protected by some form of envelope.
3. They use the cellular machinery of host cells to make copies of themselves.

While we all know that viruses can be pathogenic and dangerous (the recent example of SARS-CoV2 springs to mind), that is not necessarily the case. Some viruses like GBV-C[426] or certain strains of H5N1 *Influenza*[427] are non pathogenic and essentially harmless.

Viruses have been discovered for all three domains of life: Eukaryota, Bacteria and Archea. In Eukaryota many viruses have been discovered for animals (both vertebrate[428] and invertebrate[429]), plants,[430] protozoa,[431] chromista[432] and even fungi.[433] Bacterial viruses known as phages have been known to exists since the beginning of the 20th century.[434,435] These bacteriophages are being considered as a therapeutic alternative to antibiotics[436,437] which could help with multi-drug-resistant bacterial pathogens. Archea are also known to have their own viral infections.[438,439]

Strangely even viruses of viruses seem to exist, such as the plant satellite virus[440,441] or hepatitis delta virus.[442,443] These "viroids" do not infect viral hosts *per se* but they cannot replicate on their own. Replication must happen during co-infection with a larger virus. More recently, true viruses of viruses called virophages have been discovered. These virophages like sputnik[444] or zamilon[445] specifically infect giant viruses.

There is a huge diversity of viruses affecting all types of life, and new viruses are being discovered all the time.[446] This diversity hints at a rich and long evolutionary history. When and where viruses originated is still under study[447,448] and we might never know how they emerged. It is, however, believed that they may have played an important role in the emergence of eukaryotic cells.[449] This co-evolution between virus and host cell shows a strong link between the two organisms and some parts

of the human genome are likely of ancient viral origin.[450,451] It has been estimated that 1% to 8% of the human genome are endogenous retroviral sequences.[452,453]

The rich diversity of viruses is reflected in the variety of genetic information support, replication strategy, physical and genomic size, as well as shape. The differences in genetic information support and replication strategy form the basis of the Baltimore virus classification system[454] , still used today[455] to classify virus lineages.

As stated above, all viruses have some genetic information. This information is stored either as DNA or as RNA, which is the molecule of choice for 70% of human pathogenic viruses[456] (HIV and SARS-CoV 2 are RNA viruses).

For DNA viruses, the molecule can be double-stranded as for *Herpesvirus*,[457,458] single-stranded like in the case of *Papillomavirus*[459] or even circular in the case of the Hepatitis B virus.[460] This molecular diversity is also present in RNA viruses where the RNA molecule can be double-stranded like for *Rotavirus*,[461] or single-stranded. Furthermore, for single-stranded RNA viruses the strand can either be positive (*i.e.* can be directly translated into a protein) like the Hepatitis C virus[462] or *Poliovirus*;[463,464] conversely there are negative-strand RNA viruses, for which the complementary strand of RNA must be synthesized before translation into a protein, such as the Influenza or Measles viruses.[465]

This diversity in genetic information support implies a necessary diversity in replication strategy. The main replication strategies are as follows:[466]

- The RNA molecule is directly copied as RNA. This is the strategy followed by single-stranded RNA coronaviruses,[467] Dengue viruses[468] or Hepatitis C virus.[469]
- The DNA molecule is directly replicated as DNA. this can happen for both single-stranded DNA viruses like *Papillomavirus*[470]and double-stranded DNA viruses like Herpes simplex virus.[471]
- The DNA molecule is replicated by going through an RNA intermediary like Hepatitis B virus.[472]
- The RNA molecule is replicated by going through a DNA intermediary. This strategy is used by retroviruses that integrate this viral DNA intermediary into the host DNA, like HIV-1 (see Section 5.2.2).

Finally, the genetic diversity of viruses is reflected in their physical characteristics: viruses come in all shapes and sizes. Physical size range from 17nm for plant satellite viruses[473] to the giant, 400nm *Mimivirus*.[474] Genomic size is also quite variable. There is a stark contrast between the 860 bp *Circovirus SFBeef* and the 2.5 Mbp *Pandoravirus salinus* genomes.[475] Viruses also come in a variety of shapes:[476] icosahedral for HIV, helical for the tobacco mosaic virus or a distinctive head-tail shape for bacteriophages.

Although there are a large number of viruses, and many of them are of great importance for human health, we will now focus on one virus of particular importance: Human Immunodeficiency Virus otherwise known as HIV.

5.2. Getting to know HIV

5.2.1. Quick presentation of HIV

HIV is a single-stranded RNA retrovirus that is responsible for the Acquired Immune Deficiency Syndrome (AIDS) pandemic that has been around for the last couple decades. This virus is transmitted through sexual contact or through blood. Sexual activity is the largest transmission factor followed by intravenous drug use.[477,478]

HIV infects cells of the host immune system, specifically CD4 T-cell lymphocytes and destroys them due to its replication process.[479] CD4 T-cells are an essential part of the immune system response, helping fight against infection in humans. An HIV infection typically starts with an asymptomatic phase that can last years, followed by a growth in viral replication leading to a decrease in CD4 cells which progresses into AIDS.[480] During AIDS, when the CD4 cell count is low enough, opportunistic diseases such as pneumonia or tuberculosis[481] can easily infect the host, leading to death when the immune system is weak enough.

HIV/AIDS is one of the deadliest pandemics in history, estimated to have lead to the death of 36 million people.[482] In 2010 approximately 33 million people were infected with HIV,[483] 2.6 million of which were due to new infections, and 1.8 million died of AIDS. Most of the new infections happened in economically developing regions of the world, 70% of them coming from sub-Saharan Africa.[483] As of 2020, these numbers have decreased with "only" 1.5 million new infections and 680,000 AIDS deaths, which is encouraging from a public health perspective.

The HIV-1 virus was discovered simultaneously in 1983 by Françoise Barré-Sinoussi, Luc Montagnier[484] and Robert Gallo.[485] There exists a second HIV-2 virus discovered shortly after HIV-1,[486] it is however less transmissible than HIV-1 which is largely responsible for the global HIV/AIDS pandemic.[487] In Africa in 2006, HIV-1 infections were rising where HIV-2 were declining.[488]

While both viruses are of zoonotic origin, from transmissions of Simian Immunodeficiency Virus (SIV) from primates to humans, HIV-1 most likely originates from an SIV present in chimpanzees,[489–491] and HIV-2 from an SIV present in Sooty mangabeys.[492–494]

Several independent such transmissions have resulted in 4 lineages of HIV-1 labeled groups M, N, O and P[495] (similarly HIV-2 is split into groups A to H also resulting from independent zoonotic transmissions). Groups N and P have been identified in only a handful of individuals in Cameroon, and group O is estimated to a few

thousand cases in western Africa. The majority of the pandemic is due to viruses from group M.

The most recent common ancestor, *i.e.* the putative virus that founded group M, is estimated to have originated in what is now the Democratic Republic of Congo[496–498] at some point between 1910 and 1931.[496,499,500]

Group M is further subdivided into 9 subtypes each with distinct genetic characteristics, labeled A to K.[491,501] Like in many viruses,[502] when 2 genetically different strains of HIV co-infect a single host there is a risk of genetic recombination leading to a new strain.[503] During recombination, a new genome is formed from parts of the original genomes. This can lead to new strains that can spread and form lineages of their own. HIV strains resulting from recombination are called Circulating Recombinant Forms (CRFs). There are currently 118 identified HIV-1 CRFs in the Los Alamos National Laboratory HIV sequence database[504] (1 for HIV-2). Many unique recombinant forms (URFs) also exist. URFs and CRFs are both the result of intra-host genetic recombination a URF becomes a CRF once it has been identified in at least three epidemiologically independent infected individuals.[505] Recombination can be particularly bothersome, complicating evolutionary analyses,[506] facilitating the emergence of drug resistance and hindering vaccine development.[507]

While subtype C represented almost half of global infections from 2004 to 2007, subtype B is the majority subtype in richer countries of North America and Western Europe[508] where sequencing efforts are more common. This accounts for an over-representation of subtype B sequences in public databases such as the Los Alamos sequence database where 54% of sequences are of the B subtype and only 15% are C.[509]

5.2.2. The replication cycle of HIV

The virus's replication cycle and its immune-cell host specificity are what makes it particularly dangerous. This replication cycle can broadly be categorized into 9 separate steps[510,511] shown in Figure 5.1.

1. An HIV virion binds itself to the CD4 host cell through membrane proteins.
2. The virion envelope and host cell membrane fuse together, allowing the viral genetic material and proteins to enter the host cell.
3. The viral RNA is reverse-transcribed into viral DNA.
4. The viral DNA is integrated into the host cell genome.
5. The integrated viral DNA is transcribed by the host cell machinery into multiple copies of viral RNA.
6. The viral RNA is translated into immature viral polyproteins.

7. The viral polyproteins are cleaved to form individual viral proteins.
8. The newly synthesized viral RNA and viral proteins gather around the host-cell membrane which starts budding to create a new virion.
9. Once the budding is complete, the virion is released from the host cell and matures before being able to infect other CD4 cells and replicate again.

The successive infection of CD4 cells by HIV virions leads to cellular death due to inflammatory response and/or activation of apoptosis.[512,513] The gradual depletion of CD4 cells in the infected individual's body lead to the suppression of the immune system, and eventually to AIDS.

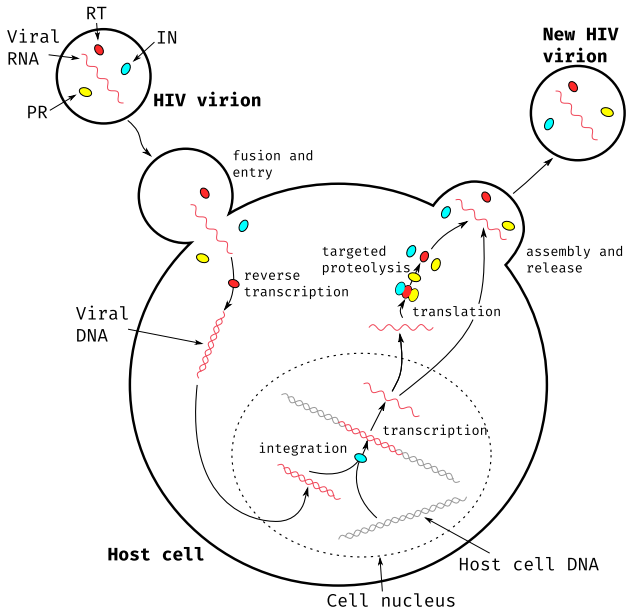


Figure 5.1.: **Main steps of HIV-1 replication cycle.**
The HIV virion contains viral RNA and three essential proteins: Reverse Transcriptase (RT) represented in red, Integrase (IN) represented in cyan and Protease (PR) represented in yellow.

5.2.3. Genetics of HIV

The replication cycle described in Section 5.2.2 is made possible by the 15 proteins of HIV. These proteins are coded by 9 separate genes.[514] An overview of the HIV proteins, their structure and localization within the viral particle can be seen in Figure 5.2.

The HIV genome is made up of three main genes each coding for polyproteins and six genes coding for proteins with regulatory or accessory roles. The three polyproteins

correspond to long chains of amino acids which are subsequently cleaved at specific positions to produce separate viral proteins.

The *gag* ("group-specific-antigen") gene codes for the Gag polyprotein which, once cleaved, results in four proteins with mainly structural roles:

- The Matrix protein (MA or p17) lines the internal surface of the virion membrane, maintaining the shape and structural integrity of the virion.
- The Capsid protein (CA or p24) forms an inner core (the capsid) inside the virion around the viral RNA. It helps protect the viral genetic information.
- The Nucleocapsid protein (NC or p7) binds with the viral RNA inside the capsid, stabilizing the molecule and further protecting the genetic information.
- The p6 protein is a small, largely unstructured protein[515] that is suspected of playing a role in virion budding and release from the host cell at the end of the replication cycle.[516,517]

The *pol* ("polymerase") gene codes for the Pol polyprotein. After cleaving, this results in three essential viral enzymes at the heart of the replication cycle:

- The Protease (PR) is responsible for cleaving the Gag, Pol and Env polyproteins to get the individual viral proteins. Without it, the individual viral proteins cannot come into being and therefore cannot function, stopping viral replication.
- The Reverse Transcriptase (RT or p51/p66) is responsible for synthesizing viral DNA from the viral RNA template contained in the virion. This is the first step in hijacking the cellular machinery for replication. Without viral DNA, HIV replication is impossible.
- The Integrase (IN) is responsible for integrating the viral DNA produced by RT in to the host cell DNA. Once the viral DNA is inside the host genome it can be transcribed and then translated (as described in Section 1.1) to produce new copies of the viral RNA and proteins. Without this integration step the viral genetic information cannot be expressed and the replication cycle is stopped.

These three proteins are of particular importance and we will go into more detail about them in Section 5.3.2.

The *env* ("envelope") gene codes for Env, the third and last polyprotein. The two resulting proteins coat the membrane of the virion and are responsible for binding with the CD4 host cells.

- The Surface protein (SU or gp120) binds to receptors on the surface of CD4 cells and allows the virion to attach itself to the host cell.[518] It also enables membrane fusion, the essential first step in the viral replication cycle.[519]

- The Transmembrane protein (TM or gp41) anchors SU into the virion membrane.

The 6 remaining genes all code for single proteins. Two of these have essential regulatory roles and the remaining four accessory roles.

The *tat* ("trans-activator of transcription") gene codes for Tat, the first essential regulatory protein. Tat activates and promotes transcription leading to more numerous and longer copies of the viral RNA.[520] The *rev* (for "regulator of virion") gene codes for Rev, the second essential regulatory protein. Rev helps transcribed viral RNA exit the nucleus of the host cell in order to be translated to viral proteins or be packaged in new, budding virions.[521]

The remaining four accessory genes are as follows: *nef* ("negative regulatory factor") code for the Nef protein which prevents the production of the CD4 cellular defense proteins increasing infectivity;[522] *vif* ("viral infectivity factor") codes for the Vif protein which also increases viral infectivity;[523] *vpu* ("viral protein U") codes for Vpu which likely helps during release of new virions[523,524] as well as preventing production of CD4 in the host cell. It is not believed to be present in the mature virion as it binds to host cellular membranes;[525] *vpr* ("viral protein R") likely helps viral DNA enter the host cell nucleus and prevents the natural host cell reproduction cycle.[526]

The existence of a 10th HIV-1 gene was suggested in 1988,[527] overlapping the *env* gene and coding for proteins on the other strand of viral DNA than the other genes. This putative gene was named *asp* ("antisense protein") and Asp transcripts were isolated during an HIV-1 infection in 2002.[528] The function of this protein is still unknown but it has been shown to have a strong evolutionary correlation with HIV-1 group M responsible for the pandemic.[529] This Asp protein is still a source of debate and is under active research.[530]

5.3. Drug resistance in HIV

Although the HIV/AIDS pandemic has been very deadly around the world, we are not completely defenseless against it. The first antiretroviral therapy (ART) drugs were made available in the late eighties, only a couple years after discovering the virus. ART reduce the viral load in an HIV positive patient reducing its transmissibility.[532] While ART is not a cure for an HIV infection it has been shown to drastically reduce mortality and morbidity.[533] ART is estimated to have saved the lives of 9.5 million individuals between 1995 and 2015.[534]

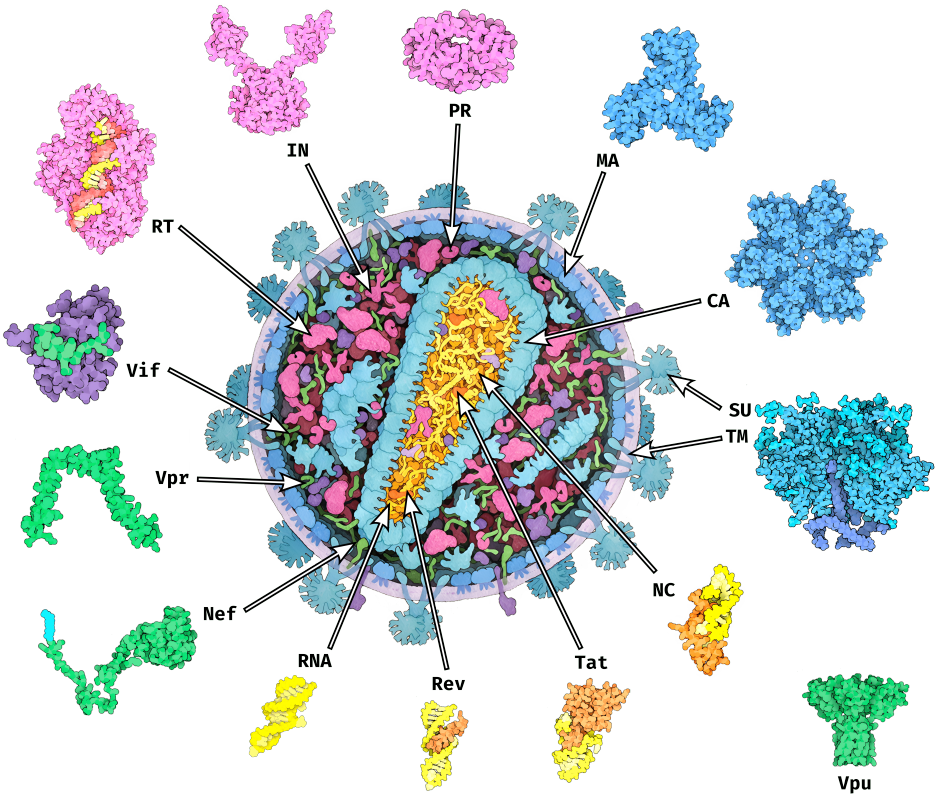


Figure 5.2.: **Structure and main components of a mature HIV-1 virion.** Structural proteins MA, CA, SU and TM are represented in Blue, functional enzymes RT, IN and PR in pink, RNA binding proteins Rev, Tat and NC in orange and accessory proteins Vif, Nef, Vpr and Vpu in green. Viral RNA is shown in yellow. The phospholipd membrane of the virion is shown in a light purple color. The p6 protein is not represented as it is largely unsctructured. Vpu is not believed to be present in the HIV virion.
Figure adapted from PDB101[531] (PDB101.rcsb.org, *CC By 4.0 License*, detailed list of structures used available in Appendix Appendix B).

5.3.1. A quick history of ART

The first available anti-HIV drug was Zidovudine (ZDV, also known as azidothymidine or AZT) approved by the FDA for usage in the USA in 1987,[535] a few years only after the discovery of the virus. This drug was a reverse transcriptase inhibitor (RTI) therefore preventing the viral RNA from being transcribed into viral DNA. Unfortunately, 3 years later, strains of HIV resistant to ZDV were circulating.[536] This rapid emergence of resistance to treatment is common for HIV[537] due to its very high evolution rate[538] allowing it to explore many possible mutations in response to selective pressures, as well as the frequent occurrence of genetic recombination.[539] To counter this resistance new drugs were rapidly developed and, between 1988 and 1995, four more RTIs were approved by the FDA. Using a combination of these drugs was also shown to be effective and led to a slower rise of resistance.[540]

Then, focus was shifted to the development of a new type of drug: Protease Inhibitors (PI). Between 1995 and 1997, 4 of them were approved. These, taken in combination with RTI made it harder for the virus to develop resistance.[541] A new class of RTIs was also explored, Non-Nucleoside RTIs (NNRTIs) that block the RT action in another manner than the previously approved Nucleoside RTIs (NRTIs). When taken in combination with other drugs they are also highly effective.[542] As the years advanced even more drug targets were explored, with 5 Integrase inhibitors (INSTI) being approved since 2007,[543] A Fusion Inhibitor (FI) in 2003,[544] and 3 other Entry inhibitors (EI)[545,546] since 2007 all targeting different steps in the replication cycle of HIV (see Table B.1 and Figure 5.3).

In response to the rapid emergence of resistance in HIV when treated with a single drug, clinicians started systematically treating HIV with a combination of multiple drugs targeting different proteins, as early as 1996. This is now referred to as highly active antiretroviral combination therapy (HAART, also known as tritherapy). HAART usually consists of 2 NRTIs coupled with another drug: NNRTI or PI at first and later FI or INSTI.[547] As of 2008, 22 anti-HIV single drugs were approved by the FDA,[548] and 27 as of today. This large array of available drugs made HAART possible and gave options to clinicians to switch targets when the multi-resistant HIV emerged. It is important to note here that, while high-income countries had access to this large panel of antiviral drugs, in most lower-income countries that was not the case. This meant that drug switching and second-line[a] drug regimens were rarely possible in these countries, leading to multi-resistant viruses.[549]

With the advent of HAART, patients had access to more potent treatments. However, the complexity of treatment regimens grew. They often involved several pills a day, taken at precise intervals. Complex drug regimens have been associated with poorer treatment adherence.[550,551] This can lead to poor treatment outcome, as well as the emergence of multi-resistant HIV strains[552] and their spread within

[a] When the anti-HIV therapy starts clinicians use first-line drug regimens, if this treatment is changed due to resistance emergence then the second-line regimen is used.

the population. To avoid this issue, increasingly more single pill regimens are being developed with a staggering 7 new drugs approved by the FDA in 2018. These single pill regimens greatly reduce the burden of adherence for patients, leading to better therapeutic outcomes, and reduced healthcare costs.[553]

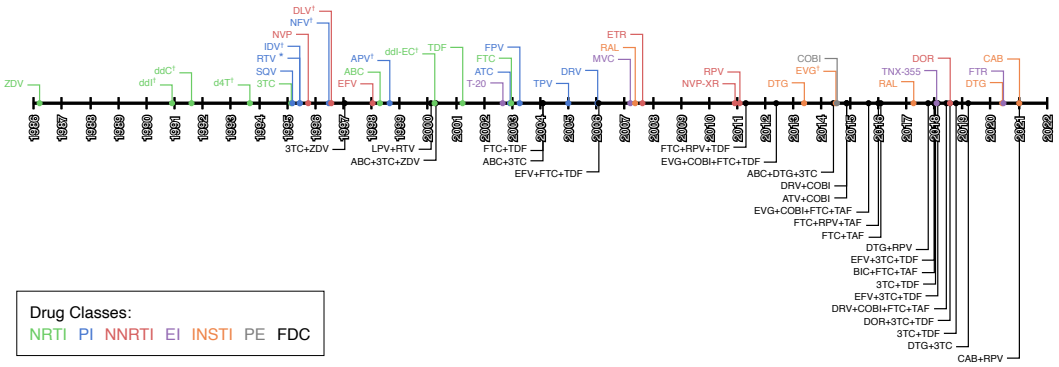


Figure 5.3.: **Timeline of ART single drug FDA approvals.**
Colored by drug type: Nucleoside Reverse transcriptase inhibitors (NRTI), Non-Nucleoside Reverse transcriptase inhibitors (NNRTI), Protease Inhibitors (PI), Integrase inhibitors (INSTI), Entry Inhibitors (EI) and pharmacokinetic enhancers (PE). Fixed Dose Combination (FDC) single pill regimens are also shown.
* RPV is often also used as a pharmacokinetic enhancer in combination with other drugs.
†These drugs are no longer approved by the FDA or no longer recommended as first line regiment treatment.
Information collected from https://hivinfo.nih.gov/understanding-hiv/fact-sheets/fda-approved-hiv-medicines, https://hivinfo.nih.gov/understanding-hiv/infographics/fda-approval-hiv-medicines and https://www.accessdata.fda.gov/scripts/cder/daf/index.cfm.
See also Table B.1.

Most recently, some studies explored using some of these single pill regimens (such as Truvada, c.f. Table B.1) as prophylactics, called Pre-exposure prophylaxis (PrEP). Putting uninfected but at risk populations on ART, before any known exposure, has been shown to effectively lower the risk of infection.[554–556] When adherence is maintained, this risk reduction has been estimated to be between 44% and 100%.[557] As of 2022, Truvada is the only authorized drug for PrEP in Europe.[558] Descovy and Apretude are also authorized for PrEP in the USA.[559]

All of these drugs are widely used and are by now very well studied, therefore detailed guidelines on all the aspects of ART; when to start, which drugs to use, when to change drugs; are issued and updated regularly by practitioners[560] and global instances[561] alike.

5.3.2. Main mechanisms of viral proteins, antiretroviral drugs and associated resistance.

Each ART drug targets a specific protein. Most of them target one of the three *pol* proteins: RT, PR and IN. The structure of these proteins is inherently linked to their function, and as such is essential to take into account when developing ART. Similarly, the structure of these proteins is very important when studying the resistance mechanisms developed by the virus.[562,563] In this section we will go over the main structural elements and how they relate to treatment and resistance, for RT, IN and PR.

5.3.2.1. Reverse transcriptase

The reverse transcriptase protein is the most targeted protein, in number of ART drugs (*c.f.* Figure 5.3 and Table B.1). The mature protein is formed of two subunits: p51 and p66. These two subunits are translated from the same section of the *pol* gene, and have the same amino acid sequence, but p51 is cleaved and is shorter than p66. The p66 subunit contains the active sites of RT whereas p51 plays a mainly structural role.

The p66 sububit can be separated into 5 domains.[564] The "fingers", "palm", and "thumb" domains are linked together and folded to form a canal through which the RNA template and newly synthesized viral DNA can pass through. The polymerase active site, responsible for incorporating nucleotides to the viral DNA molecule, is situated in the "palm" domain at the bottom of the canal. The "RNase" domain of RT contains a secondary active site responsible for cleaving the viral RNA template from the viral DNA so that the RT can fill out the complementary strand of viral DNA before integration into the host genome. The final "connection" domain is simply a link between the "RNase" and the "thumb" domains. A three dimensional view of RT with these domains highlighted can be seen in Figure 5.4.

Reverse Transcriptase inhibitors can be separated into two classes: Nucleoside RTIs (NRTIs) and Non-Nucleoside RTIs (NNRTIs). They inhibit the action of RT in two disctinct manners:

- NRTIs are analogues of free nucleotides in the host cell. They competitively inhibit RT and can be used to elongate the viral DNA chain. Once an NRTI is incorporated, further elongation of the DNA molecule is impossible and the viral DNA cannot be synthesized anymore. This is similar to the chain terminating nucleotides introduced in Section 1.2.
- NNRTIs bind to a specific region of the p51 subunit: the Non Nucleoside Inhibitor Binding Pocket (NNIBP) (A view of RT with the NNIBP visible is shown in Figure 6.4). This pocket, although it is on the p51 subunit is spatially situated very close to the polymerase active site. NNRTIs bind to the NNIBP

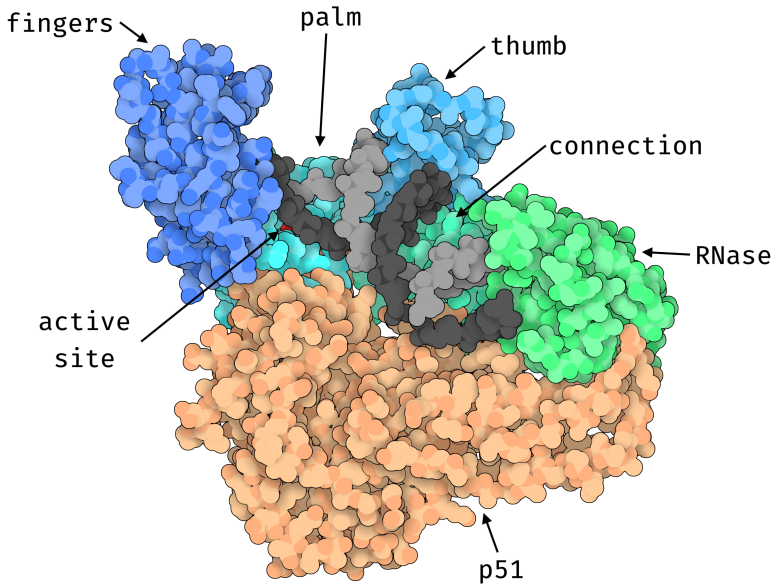


Figure 5.4.: **3D structure of HIV-1 Reverse-transcriptase.**
The different domains of the p66 subunit are labeled and shown in different shades of blue and green. The structural p51 subunit is shown in orange. The RNA template is shown in dark gray and the newly synthesized DNA strand in light gray. The polymerase active site is shown in red, although mostly hidden by the RNA template. The 3D visualization was produced with Illustrate[565] using the 2hmi PDB structure.

to change the conformation of the active site, lowering its flexibility,[566] and thus non-competitively inhibiting the action of RT.

Research has been conducted into inhibition of the RNase active site of RT[567,568] which could also inhibit the action of RT. There is, however, to this day, no approved treatment that inhibits the RNase action of RT.

Drug resistance mutations (DRMs) that arise in HIV from the selective pressures resulting from RTI exposure can similarly be grouped into two categories: NRTI and NNRTI resistance mutations.

NRTI resistance mutations can further be subcategorized into two groups.[569,570] The first type of NRTI resistance mutations are mutations that prevent the incorporation of NRTIs into the viral DNA molecule. M184V and M184I, indicating the replacement, at site number 184, of a Methionine by a Valine or an Isoleucine respectively, are very common NRTI resistance mutations. These V and I amino acids have a different structure than the original M, interfering with the incorporation of lamidu-vine (3TC) but not dNTP.[571] The second type of mutation, allows RT to remove an incorporated NRTI from the viral DNA to resume synthesis. Thymidine Analog Mutations (TAMs), M41L, D67N, K70R, L210W, T215Y/F and K219Q/E confer resistance to azidothymidine (AZT) through this mechanism.[572,573]

Similarly, NNRTI resistance mutations work via several different mechanisms.[574,575] Some NNRTI resistance mutations, like Y181C, lower the affinity of the NNIBP to NNRTIs preventing binding of drugs to RT. Others, like K103N change the conformation of the p51 subunit, making the NNIBP disappear. NNRTI resistance mutations are particularly dangerous because they often confer cross-resistance to multiple NNRTIs without affecting the polymerase action very much,[562] giving rise to viruses that are both fit and highly resistant. This is contrast to NRTI resistance mutations that generally incur a fitness cost for the virus, lowering its efficacy.[576]

5.3.2.2. Protease

The Protease protein, also a major drug target for ART, cleaves the *gag* and *pol* polyproteins in order to produce functional viral proteins, essential to replication. It has a symmetric, dimeric, structure. That is to say: it is composed of two identical chains of amino acids.[577,578] A structural view of PR is shown in Figure 5.5.

These two chains are folded in order to create a "tunnel" through which the polyproteins enter. In the middle of this "tunnel", at the bottom, is the active site. The active site is composed of two Aspartate residues, one on each chain. Using water, they can provoke a chemical reaction that cleaves the polyprotein at a specific position.[579]

The roof of the "tunnel" is formed by the flaps, a flexible region from each of the two chains that can open or close the "tunnel".[580] These flaps most likely control the access of polyproteins to the active site.[581,582]

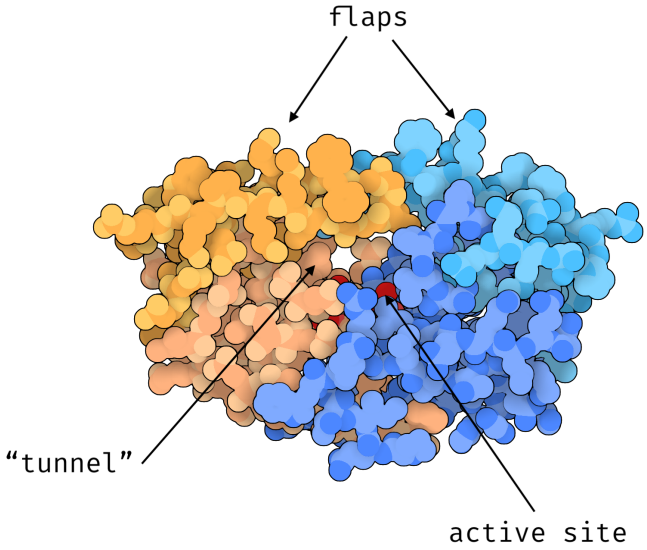


Figure 5.5.: **3D structure of HIV-1 Protease.**
The two identical chains are colored in orange and blue shades respectively. The flexible flaps form the the "roof" of a tunnel, at the bottom of which is the active site: 2 Asp residues, one on each chain. The 3D visualization was produced with Illustrate[565] using the 2p3b PDB structure.

All the approved Protease Inhibitors (PIs) share a similar mode of action. Each PI binds to the active site of the PR, denying access to the "tunnel" for polyproteins, and stopping the catalytic action of PR.[583,584] Tipranavir, one of the more recent PIs, also binds with the flaps.[584]

According to Prabu-Jeyabalan *et al.*, PR does not recognize the specific sequence of the polyprotein cleavage site but rather its shape.[585] They proposed an inhibitor based on the shape of all polyproteins combined, which establishes more bonds with PR, making it supposedly more efficient[586] than current approved PIs.

As is the case with RTIs, when under selective pressure due to PIs, the virus tends to develop PI associated DRMs. Most PI resistance mutations result in an enlarged "tunnel". This tends to lower the affinity of the PIs to the active site, but also the affinity of polyproteins, lowering the fitness of the virus significantly.[541] In addition, some mutations on the *gag* polyprotein seem to lower the efficacy of PIs, although the underlying mechanism is not well known.[541]

Some mutations in the flaps of PR have also been shown to confer PI resistance. It seems likely that these mutations change conformation of the flaps, opening them and leading to the release of inhibitors from the active site.[587]

5.3.2.3. Integrase

The integrase protein is the third major anti-retroviral drug target. It is responsible for integrating the viral DNA into the host genome. IN is a tetramer composed of four identical amino acid chains.[588,589] Each of these chains contain three domains linked together by flexible linker sequences: the N-terminal domain, the catalytic core and the C-terminal domain. In each tetramer, two chains provide the active site for the integration reaction while the other two have a mostly structural role. It is probable that the N-terminal domain, which is very conserved, is necessary for stable tetramerization of IN monomers.[590] This tetrameric structure is shown in Figure 5.6.

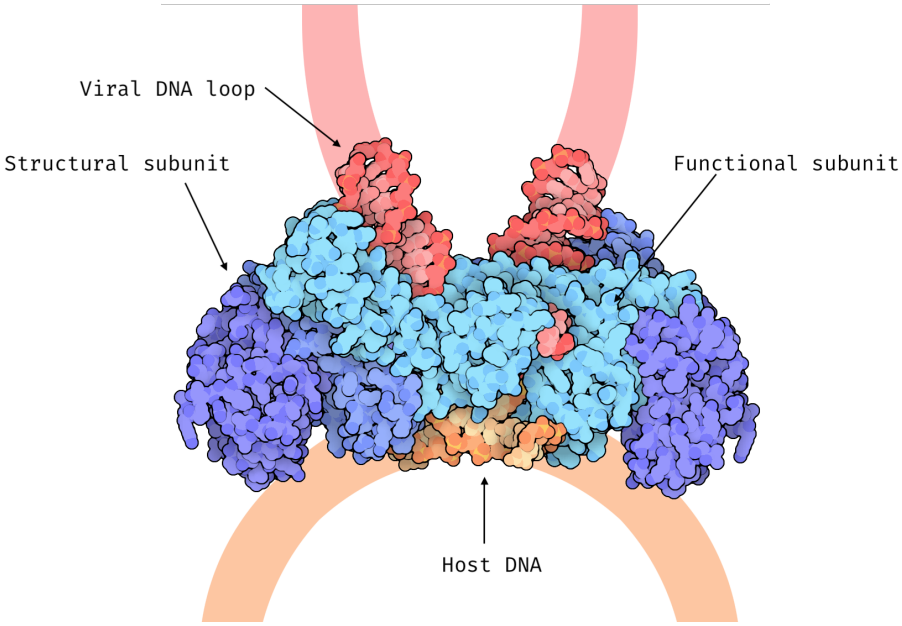


Figure 5.6.: **3D structure of an Integrase.**
This Integrase tetramer is binded with viral (red) and host (orange) DNA, linked to the two light blue functional subunits via the C-terminal domain. The active site formed by the the catalytic cores of the two functional subunits *(not visible in this representation)*, is where the strand transfer reaction will take place. The two dark blue IN subunits have a structural role. This figure was adapted from the PDB 101 molecule of the month Integrase entry by David S. Goodsell and the RCSB PDB (https://pdb101.rcsb.org/motm/135) with a CC By 4.0 license.

Several steps are needed in order to integrate the viral DNA with the host genome.[591] First, IN binds to the both ends of the viral DNA, using the C-terminal domains, forming a closed loop. Secondly, both ends of the viral DNA molecule are then prepared for integration by the catalytic core. Third, the host DNA is captured with C-terminal domains. Then, the strand-transfer is done within the catalytic core: the host DNA is cut in two places and a single strand from each end of the viral DNA are attached to these two breakpoints. Finally, the IN tetramer detaches from the linked molecules and the final steps necessary to create a single hybrid DNA molecule are done by the host cellular machinery. A graphical representation of this process can be found in Figure 1 of Maertens *et al.* (2022).[591]

Integrase Strand Transfer Inhibitors (INSTIs), as their name indicates, block the strand transfer reaction. They achieve this by strongly binding to the active site of the IN tetramer after it has formed a complex with the viral DNA.[591,592] In doing so, INSTIs prevent the IN / viral DNA complex from binding to the host DNA, effectively preventing strand transfer.

In the presence of INSTIs during therapy, once more, the HIV virus develops resistance mutations over time. These mutations all lower affinity of IN to INSTIs, preventing bonding.[591,593] Since most INSTIs behave similarly, this means that cross-resistance to INSTIs is quite common for INSTI DRMs.[593,594] Again, these mutations tend to lower the overall viral fitness necessitating secondary compensatory mutations to restore fitness.[593,594]

5.3.2.4. Other drug targets

For now, resistance has not been observed for novel drugs like entry inhibitors. This might be because the genetic barrier to resistance is higher and not enough time has passed since their introduction for resistance to emerge.

For all the other drug targets however, as stated earlier in this section, resistance is documented and problematic. Resistance has even been detected for PrEP which is prophylactic.[595,596] This seems to be rare however, and mostly due to unknown pre-treatment HIV infections.[597]

5.3.3. Consequences of resistance on global health

HIV resistance to ART drugs is problematic from a global health perspective. Indeed, circulation of resistant strains of HIV within populations can lead to treatment-naive individuals that will not respond well to treatment.

More concerning, is the fact that transmission of resistant strains of HIV between treatment-naive individuals is the main mode of resistance transmission in the UK[598,599] and Switzerland.[600] This treatment-naive to treatment-naive transmission

is particularly insidious since it can go undetected and creates long lasting drug resistant strain reservoirs in the treatment-naive population. This of course is dangerous since some infected individuals might experience poor therapeutic outcomes and even treatment failure when administered first line regimens.[601] To avoid this, genotypic resistance testing has become standard practice when choosing the therapeutic strategy in high-income countries, but more effort must be done to make resistance testing less expensive and more cost-efficient in lower and middle income countries.[602]

Although the transmitted drug resistance described above is problematic, a large portion of DRMs incur a fitness cost for the resistant strain.[603,604] This means that, although they are selected when exposed to the evolutionary pressure of ART, when the treatment is interrupted there is another pressure leading these costly mutations to disappear. This reversion is commonly observed after interruption of treatment, however the median reversion times vary widely from 1 to 13 years[605] depending on the severity of the fitness loss and type of mutation. This means that, although reversion can possibly lead to loss of resistance, this can potentially take a long time and possibly longer than the treatment interruption.

In practice, it is therefore very important to keep an eye on all drug resistance mutations, their population dynamics, and spread as well as their presence or absence in a particular strain before starting treatment.

5.3.4. Finding DRMs [b]

Finding and categorizing mutations as DRMs is an important task in light of the public health implications mentioned in Section 5.3.3. As such, this is an active part of the HIV research field.

The most important thing needed in order to study DRMs is, of course, viral sequences. To facilitate the search for DRMs, several sequence databases exist. Sequences are often linked to metadata related to the treatment status of the patient from which the sequence was obtained. This metadata can be quite variable: from a coarse level binary indicator of treatment to a finely detailed list of all treatments received and associated phenotypic measurements like viral load.

Databases like the UK-CHIC,[607] UK HIV drug resistance database (`https://www.hivrdb.org.uk/`) and Swiss cohort study (`https://www.shcs.ch/`) host sequences on a national level, although access can be granted to international researchers. Other databases like the PANGEA database[608] host sequences from multiple countries in sub-Saharan Africa. The Stanford HIV drug resistance database (`https://hivdb.stanford.edu/`) hosts HIV sequences with some phenotypic data.[33,609] Finally some database only host sequences, such as the Los Alamos HIV

[b] This sections build upon a review I participated in during my PhD[606]

sequence database (http://www.hiv.lanl.gov/). However, with few specific treatment or resistance related metadata,[610] these have less direct applicability to the DRM search task.

Some databases, like the Stanford HIV resistance database also store specific knowledge about known resistance mutations, keeping and regularly updating lists of clinically important DRMs as well as their impact on ART.[611,612] Additionally, Stanford also offers tools for clinicians to do genotypic resistance testing with interpretable results.[613]

The first step of mutations discovery is usually some kind of statistical association analysis[611,614] where the association between treatment status (coarse of fine grained) and specific mutations is statistically tested. This is usually done with Fisher association tests[615,616] or correlation testing with the Spearman correlation.[617] This results in a list of mutations that are significantly associated with a given treatment and corresponding p-values.

Since, on a given sequence dataset, several mutations are usually tested at once, this can lead to inflated false positives[618] and spurious associations.[619] Fortunately, this is a well studied problem and many methods exist to control this effect by controlling the Familywise Error Rate (FWER) *e.g.* with the Bonferroni procedure,[620] or the False Discovery Rate (FDR) *e.g.* with the Benjamini-Hochberg procedure.[621] These methods are often applied when testing for resistance association.[615,622,623] However, these correction methods are a double-edged sword, some of them can be very conservative and lead to falsely rejecting true associations.[624] In some studies on resistance, phylogenetic correlation between the sequences is also accounted for.[625,626]

Statistical testing on treatment status, while informative, can only associate a mutation with a treatment. In order to actually validate whether a mutation causes resistance or not, biological analyses are needed.[611,614] The easiest of these are *in vitro* analyses where live viruses are subjected to a phenotypical assay. These assays measure the susceptibility of HIV viruses to a wide array of drugs, which can then be statistically associated with genetic traits like specific mutations. These assays like phenosense[627] or antivirogram[628] are widely used.[629–631] Viruses can be obtained from clinical isolates,[632] or viruses with specific mutations can be manufactured with site directed mutagenesis.[633,634] *In vivo* studies can be conducted by sequencing viruses from patients failing ART, following over time and studying the association between their treatment response and HIV genetics.[635,636]

More recently, as sequence database grow bigger and bigger *(The UK-CHIC database contains more than 80,000 HIV sequences with treatment status)*, methods based on statistical and machine learning are being used to study resistance. Most approaches rely on training models to predict some type of resistance: either classifying

sequences as resistant or not[331,637] of predicting a phenotypic response like fold resistance compared to wild type.[638] Initial approaches were mainly designed for clinical testing, rather than new DRM search, and distributed via web services.[639,640]

Initially these approaches were based on models like decision trees,[641] SVMs[639] or logistic regression.[642] Over time the use of more complex models such as neural networks has increased, with increased prediction accuracy.[638]

By analyzing the important features used by trained models to predict resistance, it is possible to find features corresponding to mutations, that are useful for predicting, and therefore likely associated with, drug resistance (see Chapter 6). With the improvement in methods to interpret and extract features from complex models such as deep neural networks, this approach has been used with deep learning models.[331] This novel way of finding resistance associated mutations has the potential to uncover complex mutational effects that simple association testing cannot.

5.4. Conclusion

Viruses are surprisingly complex in light of their apparent simplicity. They are ubiquitous and present an extreme diversity. Whether they are pathogenic or not, the role of viruses in a myriad of processes and niches make them interesting and important to study. The sequences of these viruses, although small can be very useful for evolutionary as well as clinical analyses.

Although the study of viruses as a whole is very useful, HIV is particularly important to study. The impact of the HIV pandemic on global health has been severe, both in Lower and Higher income countries. It is therefore paramount to fully understand the underlying mechanisms and evolutionary adaptations of this virus. Its high mutation rate allows it to quickly explore evolutionary alternatives when exposed to drugs, making anti HIV therapy a complex endeavor.

Fortunately, with large scale sequencing efforts it is possible to study and track these evolutionary adaptations to treatments. This allows us to adapt therapeutic strategies as well as develop new compounds and approaches. In this context, studying and finding the virus's mutational processes is paramount. This is especially important when studying resistance to RTIs as they form the backbone of first line regimen combination therapies, and are the most common type of anti-HIV drug. This process is made easier by the large scale sequence repositories now available, and the usage of machine and statistical learning to leverage that data.

6. Contribution 2: Inferring Mutation Roles From Sequence Alignments Using Machine Learning

As we have seen in Sections 5.2.1 and 5.3.3, the HIV pandemic is a widespread threat to public health which can have very serious consequences at the infected individual's level and at the population scale. Despite many advances in drug development, fundamental research in new drug targets drug resistance mutations arise very quickly in response to antiretroviral therapy. This is especially important in lower income countries where the drug switching options are less numerous and give rise to multi-resistant viruses. In order to manage this global pandemic, surveying the viral infections, finding and categorizing new drug resistance mutations is very important. One such way to do this is to use HIV viral sequences, obtained from patients, and use their alignment as input for statistical and machine learning methods *(c.f. Chapter 4 and Section 5.3.4)*. In this chapter, I will present some work done on studying drug resistance mutations in HIV sequences using a large sequence alignment and machine learning methods.

This chapter was written as an article titled: **"Using Machine Learning and Big Data to Explore the Drug Resistance Landscape in HIV"**. It was originally published in *August 2021*, in *PLoS Computational Biology* (*doi:10.1371/journal.pcbi.1008873*) and is presented as is, without any modification from the published version. The author list, complete with affiliations is given below:

Luc Blassel[1,2*], Anna Tostevin[3], Christian Julian Villabona-Arenas[4,5], Martine Peeters[6], Stéphane Hué[4,5], Olivier Gascuel[1,7#] On behalf of the UK HIV Drug Resistance Database[^]

1 Unité de Bioinformatique Évolutive, Institut Pasteur, Paris, France
2 Sorbonne Université, Collège doctoral, Paris, France
3 Institute for Global Health, UCL, London, UK
4 Department of Infectious Disease Epidemiology, London School of Hygiene and Tropical Medicine, London, UK

5 Centre for Mathematical Modelling of Infectious Diseases, London School of Hygiene and Tropical Medicine, London, UK
6 TransVIHMI (Recherches Translationnelles sur VIH et Maladies Infectieuses), Université de Montpellier, Institut de Recherche pour le Développement, INSERM, Montpellier, France
7 Institut de Systématique, Evolution, Biodiversité (ISYEB), UMR 7205 - Muséum National d'Histoire Naturelle, CNRS, SU, EPHE and UA, Paris, France

Current address: Institut de Systématique, Evolution, Biodiversité (ISYEB), UMR 7205 - Muséum National d'Histoire Naturelle, CNRS, SU, EPHE and UA, Paris, France
* luc.blassel@pasteur.fr (LB)
* olivier.gascuel@mnhn.fr (OG)

^ Membership list can be found in the acknowledgments section

Abstract

Drug resistance mutations (DRMs) appear in HIV under treatment pressure. DRMs are commonly transmitted to naive patients. The standard approach to reveal new DRMs is to test for significant frequency differences of mutations between treated and naive patients. However, we then consider each mutation individually and cannot hope to study interactions between several mutations. Here, we aim to leverage the ever-growing quantity of high-quality sequence data and machine learning methods to study such interactions (i.e. epistasis), as well as try to find new DRMs.

We trained classifiers to discriminate between Reverse Transcriptase Inhibitor (RTI)-experienced and RTI-naive samples on a large HIV-1 reverse transcriptase (RT) sequence dataset from the UK ($n \approx 55,000$), using all observed mutations as binary representation features. To assess the robustness of our findings, our classifiers were evaluated on independent data sets, both from the UK and Africa. Important representation features for each classifier were then extracted as potential DRMs. To find novel DRMs, we repeated this process by removing either features or samples associated to known DRMs.

When keeping all known resistance signal, we detected sufficiently prevalent known DRMs, thus validating the approach. When removing features corresponding to known DRMs, our classifiers retained some prediction accuracy, and six new mutations significantly associated with resistance were identified. These six mutations have a low genetic barrier, are correlated to known DRMs, and are spatially close to either the RT active site or the regulatory binding pocket. When removing both known DRM features and sequences containing at least one known DRM, our classifiers lose all prediction accuracy. These results likely indicate that all mutations directly conferring resistance have been found, and that our newly discovered DRMs

are accessory or compensatory mutations. Moreover, apart from the accessory nature of the relationships we found, we did not find any significant signal of further, more subtle epistasis combining several mutations which individually do not seem to confer any resistance.

Author summary

Almost all drugs to treat HIV target the Reverse Transcriptase (RT) and Drug resistance mutations (DRMs) appear in HIV under treatment pressure. Resistant strains can be transmitted and limit treatment options at the population level. Classically, multiple statistical testing is used to find DRMs, by comparing virus sequences of treated and naive populations. However, with this method, each mutation is considered individually and we cannot hope to reveal any interaction (epistasis) between them. Here, we used machine learning to discover new DRMs and study potential epistasis effects. We applied this approach to a very large UK dataset comprising $\approx 55,000$ RT sequences. Results robustness was checked on different UK and African datasets.

Six new mutations associated to resistance were found. All six have a low genetic barrier and show high correlations with known DRMs. Moreover, all these mutations are close to either the active site or the regulatory binding pocket of RT. Thus, they are good candidates for further wet experiments to establish their role in drug resistance. Importantly, our results indicate that epistasis seems to be limited to the classical scheme where primary DRMs confer resistance and associated mutations modulate the strength of the resistance and/or compensate for the fitness cost induced by DRMs.

6.1. Introduction

Drug resistance mutations (DRMs) arise in Human Immunodeficiency Virus-1 (HIV-1) due to antiretroviral treatment pressure, leading to viral rebound and treatment failure.[643,644] Furthermore, drug-resistant HIV strains can be transmitted to treatment-naive individuals and further spread throughout the population over time.[598,599,645] These transmitted resistant variants limit baseline treatment options and have clinical and public health implications worldwide. Almost all drugs to treat HIV target the reverse transcriptase (RT), encoded by the *pol* gene. Lists of DRMs are regularly compiled and updated by experts in the field, based on genotype analyses and phenotypic resistance tests or clinical outcome in patients on ART.[646–648] However, with the developement of new antiretroviral drugs that target RT but also other regions of the *pol* gene like protease or integrase, and the use of anti-retrovirals in high risk populations by pre-exposure prophylaxis (PREP),

it is important to further our understanding of HIV polymorphisms and notably the interactions between mutations and epistatic effects.

Among known DRMs, some mutations, such as M184V, directly confer resistance to antiretrovirals, more precisely the commonly used NRTI, 3TC (lamivudine) and FTC (emtricitabine), and are called primary or major drug resistance mutations, while some mutations like E40F have an accessory role and increases drug resistance when appearing alongside primary DRMs. Moreover, some mutations like S68G seem to have a compensatory role, but are not known to confer any resistance nor modulate resistance induced by primary DRMs. All of these mutations might have different functions in the virus, but they are all known to be associated with drug resistance phenomena. Therefore, during the rest of this article we will refer to all of these known mutations as resistance associated mutations (RAMs), rather than DRMs which is too specific, and our goal will be to search for new RAMs and study the interactions between known RAMs and the new ones.

Classically, new RAMs have been found using statistical testing and large multiple sequence alignments (MSA) of the studied protein.[615,649] Tests are performed for mutations of interest on a given MSA to check if they are associated with the treatment status and outcome of the individual the viral sequences were sampled from. The test significance is corrected for multiple testing as all mutations associated to every MSA position is virtually a resistance mutation and tested. After this preliminary statistical search, the selected mutations are scrutinized to remove the effects of phylogenetic correlation (i.e. typically counting two sequences which are identical or closely related due to transmission rather than independent acquisition twice[650]) and check that the same mutation occurred several times in different subtypes and populations being treated with the same drug. Then, these mutations can be further experimentally tested in vitro or in vivo to validate phenotypic resistance. This method has worked well, but by design it is not ideal for studying the effect of several mutations at once, since if we have to test all couples or triplets of mutations, we quickly lose statistical power when correcting for multiple testing,[624] due to the large number of tests to perform. Moreover, phylogenetic correlation is again a critical issue with such an approach.

Machine learning has been extensively used to predict resistance to antiretrovirals from sequence data. There are two main approaches to predicting resistance from sequence data. Regression, where machine learning models are trained to predict the value of a drug resistance indicator, typically IC_{50} fold change in response to a given drug[651] or other indicators from phenotypic resistance assays such as PhenoSense.[652] Many methods have been used to predict a resistance level: Support Vector Machines (SVMs),[653] k-Nearest Neighbors (KNN) and Random Forests (RFs),[654] and more recently Artificial Neural Networks (ANNs).[638,655] Alternatively, this task has also been approached as a classification problem. Given a certain threshold on a phenotypic resistance measure, sequences are given a label of ”resistant” or ”susceptible” to a certain drug. Machine learning classifiers are then trained to predict that

label. For this task, SVMs and decision trees have been used,[639,656] ensemble classifier chains[642,657] and also ANNs.[658] Most recently Steiner *et al.*[331] have used Deep Learning Architectures to predict resistance status (i.e. classification) from sequence data. Since phenotypic assays are more complicated and costly to perform than simple genotyping, there is a limited number of sequences paired with a resistance level. This is the main limitation of these studies since machine learning methods typically benefit from a large amount of training data. This is especially true for deep neural networks which can need hundreds of thousands of training samples for certain tasks and architectures. However, despite this limitation, approaches proposed in these studies seem to have fairly good predictive accuracy. It is important to note that all of these studies aim to predict if a given sequence is resistant or not to a given drug, they do not aim to find new potential RAMs. Although Steiner *et al.*[331] have checked that known DRM positions are captured by their models and found several positions potentially associated to resistance, it is not the main goal of their method.

It is accepted in machine learning that there is a trade-off between model accuracy and model interpretability. In these previous studies the goal was to make the most accurate predictions possible, using complex models such as SVMs and ANNs, therefore sacrificing interpretability. Here, we have a different approach, using simpler models that might be less accurate but whose predictions we can understand and interpret. We train these models to discriminate RTI-naive from RTI-experienced sequences. Without the need for phenotypic data, we are able to use much larger HIV-1 RT sequence datasets from the UK ($n \approx 55,000$) (`http://www.hivrdb.org.uk/`) and Africa ($n \approx 4,000$).[615] By using interpretable models, we can extract mutations that are important for determining if a sequence is treated or not and potentially find new mutations potentially associated to resistance. Furthermore, we aim to detect associations between mutations and their effect on antiretroviral resistance in order to study potential underlying epistasis. The African and UK datasets are very different both from genetic and treatment history standpoints, therefore training classifiers on the UK dataset and testing them on the African one, should guarantee the robustness of our findings and greatly alleviate phylogenetic correlation effects. In the following sections, we first describe the data then the methods used. Our results include the assessment of the performance of our classifiers even when trained on data devoid of any known resistance-associated signal; as well as a description of the main features (prevalence and correlation to known mutations, genetic barrier and structural analysis) of six potentially resistance associated mutations, newly discovered thanks to our approach. These results and perspectives are discussed in the concluding section.

6.2. Materials and methods

6.2.1. Data

In this study, we used all the drug resistance mutations that appeared in the Stanford HIV Drug resistance database, both for NRTI (Nucleoside Reverse Transcriptase Inhibitors; https://hivdb.stanford.edu/dr-summary/comments/NRTI/) and NNRTI (Non Nucleoside RTI; https://hivdb.stanford.edu/dr-summary/comments/NNRTI/) as known RAMs. To discover new RAMs, assess their statistical significance and study potential epistatic effects, we used two datasets of HIV-1 RT sequences. A large one ($n = 55,539$) from the UK HIV Drug Resistance Database (http://www.hivrdb.org.uk/) and a smaller ($n = 3,990$) one from 10 different western, eastern and central African countries.[615] In the UK dataset, sequences from RTI-naive individuals formed the majority class with 41,921 sequences (75%). In the African dataset, both classes were more balanced with 2,316 RTI-naive sequences (58%). In the UK dataset, RTI-naive sequences had at least one known RAM in 25% of cases, most likely due to transmissions to naive patients or undisclosed treatment history, against 48% in RTI-experienced sequences, thus making the discrimination between the RTI-experienced and RTI-naive sequences particularly difficult. In the African dataset this distribution was more contrasted, with only 14% of RTI-naive sequences having at least one known RAM, versus 83% of RTI-experienced sequences. The African dataset was also much more genetically diverse with 24 different subtypes and CRFs compared to the 2 subtypes (B and C) that we retained for this study from the UK cohort. The majority of the sequences from the African dataset were samples from Cameroon (27%), Democratic Republic of Congo (17%), Burundi (15%), Burkina Faso (13%) and Togo (11%).

It is important to note that RTI-experienced sequences in both of these datasets can be considered as resistant to treatment. Since the viral load was sufficiently high to allow for sequencing of the virus, we can consider that the ART has failed. However, in some cases this resistance might be caused by non adherence to ART, rather than by the presence of RAMs, therefore adding some noise to the relationship between treatment status and resistance.

In addition to differences in size, balance between RTI-naive and experienced classes, and the genetic difference between the UK and African datasets, there are also significant differences resulting from differing treatment strategies. In the UK and other higher income countries, the treatment is often tailored to the individual with genotype testing, which result in specific treatment as well as thorough follow-ups and high treatment adherence. In the African countries of the dataset that we used, the treatment is ZDV/ d4T (NRTI) + 3TC (NRTI) + NVP/EFV (NNRTI) in most cases,[615] and this treatment is generalized to the affected population, with poorer follow-up and adherence than in the UK. This discrepancy could lead to different mutations arising in both datasets, however since the treatment strategy is

		UK		Africa	
size		55539		3990	
RTI naive	with known RAMs	11429	(21%)	318	(8%)
	without known RAMs	30492	(55%)	1998	(50%)
RTI experienced	with known RAMs	6633	(12%)	1388	(35%)
	without known RAMs	6985	(13%)	286	(7%)
sequences with ≥ 2 known RAMs		8034	(14%)	1308	(33%)
max known RAM number		13		17	
Median known RAM number		1		3	
number of subtypes / CRFs		2		24	
subtypes / CRFs	A	0	(0%)	472	(12%)
	B	37806	(68%)	64	(2%)
	C	17733	(32%)	702	(18%)
	CRF02 AG	0	(0%)	1477	(37%)

Table 6.1.: **Summary of the UK and African datasets.**
Percentages are computed with regards to the size of the considered dataset (e.g. 21% of the sequences of the UK dataset are RTI-naive and have at least one known RAM). The median number of RAMs was computed only on sequences that had at least one known RAM.

a combination of both NRTI and NNRTI drug classes, as in many countries, similar RAMs arise.[615] Furthermore, there is potentially more uncertainty in the African dataset than in the UK. For example some individuals may have unofficially taken antiretroviral drugs, but still identify themselves as RTI-naive, or report having some form of ART while not having been treated for HIV.[659] All of this explains the high prevalence of multiple resistance in the African data set: the median number of RAMs in sequences containing at least one RAM is 3 in the African sequences, while it is 1 in UK sequences (Table 6.1). Thus, we can say that African sequences are highly resistant, with possibly different mutations and epistatic effects, compared to their UK counterparts.

All these differences between the two datasets helped us to assess the generalizability of our method and the robustness of the results. That is to say, if signal extracted from the UK dataset was still relevant on such a different dataset as the African one, we could be fairly reassured in regard to the biological and epidemiological relevance of the observed signal.

Sequences in both African and UK datasets were already aligned. In order to avoid overly gappy regions of our alignment we selected only positions 41 to 235 of RT for our analysis. We used the Sierra web service (https://hivdb.stanford.edu/p

age/webservice/) to get amino acid positions relative to the reference HXB2 HIV genome. This allowed us to determine all the amino acids present at each reference position in both datasets, among which we distinguished the "reference amino acids" for each position, corresponding to the B and C subtype reference sequences obtained from the Los Alamos sequence database (http://www.hiv.lanl.gov/). All the other, non-reference amino acids are named "mutations" in the following, and the set of mutations was explored to reveal new potential RAMs.

To train our supervised classification methods,[386,660,661] the sequence data needed to be encoded to numerical vectors. A common and intuitive way to do so is to create a single feature in the dataset for each position of the sequence to encode. Each amino acid is then assigned an integer value, and an amino acid sequence is represented by a succession of integers corresponding to each amino acid. There is, however, one drawback with this method: by assigning an integer value to amino acids, we transform a categorical variable into an ordinal variable. Any ordering of amino acids is hard to justify and might introduce bias. To avoid this, we represented each sequence by a binary vector using one-hot encoding. For each position in the sequence to be encoded, amino acids corresponding to mutations are mapped to a binary vector denoting its presence or absence in the sequence. For example, at site 184, amino acids M, G, I, L, T and V are present in the UK dataset. After encoding we will have 5 binary features corresponding to the M184G, M184I, M184L, M184T and M184V mutations. We did not encode the reference amino acid M, but only the mutated amino acids. With this method each mutation in the dataset ($n = 1,318$) corresponds to a single feature. Some of these features corresponded to known RAMs (e.g., M184I and M184V) and are named (known) RAM features in the following ($n = 121$). This encoding allows the classifiers to consider specific mutations and potentially link them to resistance.

6.2.2. Classifier training

In order to find new potential RAMs, we first followed the conventional multiple testing approach.[615] We first used Fisher exact tests to identify which of these mutations were significantly associated with anti-retroviral treatment. All the resulting p-values were then corrected for multiple testing using the Bonferroni correction.[662] Those for which the corrected p-value was ≤ 0.05 were then considered as significantly associated with treatment and potentially implicated in resistance.

This method was complemented by our parallel, machine learning based approach. In order to extract potential RAMs, we trained several classifiers to discriminate between RTI-experienced and RTI-naive sequences represented by the binary vectors described above. This classification task does not need any phenotypic resistance measure, allowing us to use much larger and more readily available datasets than other machine learning based approaches previously mentioned. Once the classifiers

were trained, we extracted the most important representation features, which corresponded to potentially resistance-associated mutations (PRAM in short). To this aim we chose three interpretable supervised learning classification methods so as to be able to extract those features:

1. Multinomial naive Bayes (NB), which estimates conditional probabilities of being in the RTI-experienced class given a set of representation features;[663] the higher (≈ 1.0) and the lower (≈ 0) conditional probabilities correspond to the most important features.
2. Logistic regression (LR) with L1 regularization (LASSO)[386] which assigns weights to each of the features, whose sign denotes the importance to one of the 2 classes, and whose absolute value denotes the weight of this importance.
3. Random Forest (RF) , which has feature importance measures based on the Gini impurity in the decision trees.[393]

Interpretability was the main driver behind our classification method choice, with the conditional probabilities of NB, the weight or LR and the importance values of RF, we can easily extract which mutations are driving the discrimination of RT sequences. This is why we did not choose to use ANNs which could have led to an increase in accuracy at the cost of interpretability.[355,664,665] Moreover, these three classification methods have the potential to detect epistatic effects. With RF, the discrimination is based on the combination of a few features (i.e. mutations), while with LR the features are weighted positively or negatively, thus making it possible to detect cumulative effects resulting from a large number of mutations, which individually have no discrimination power. Naive Bayes is a very simple approach, generally fairly accurate, and in between the two others in terms of explanatory power.[661]

In order to be able to compare all these approaches in a common framework, we devised a very simple classifier out of the results of the Fisher exact tests. This "Fisher classifier" (FC) predicts a sequence as RTI-experienced if it has at least one of the mutations significantly associated to treatment. In this way, we were able to compute metrics for all classification methods and compare their performance.

It is important to note that in all of these approaches we chose to discriminate RTI-naive from RTI-experienced sequences, regardless of the type of RTI received. One of the reasons is that we did not have detailed enough treatment history for sequences in the UK and African datasets. Moreover, even without segmenting by treatment type, the size of the training set and the power of our classification methods were both high enough to be able to detect all kinds of resistance associated mutations. We shall see (Result section) that we were able to determine the likely treatment involved by further examining the important extracted features and comparing them to known RAMs. Furthermore, since the treatment strategies are so different between the UK and African sequences, training on sequences having received different treatments

Signal removal level	Trained on		Tested on	
None	UK, subtype B	(37806)	UK, subtype C	(17733)
	UK, subtype C	(17733)	UK, subtype B	(37806)
	UK, subtypes B & C	(55539)	Africa, all subtypes	(3990)
Known RAM features removed	UK, subtype B	(37806)	UK, subtype C	(17733)
	UK, subtype C	(17733)	UK, subtype B	(37806)
	UK, subtypes B & C	(55539)	Africa, all subtypes	(3990)
Known RAM features & sequences with $\geq$ 1 known RAM removed	UK, subtype B	(24422)	UK, subtype C	(13055)
	UK, subtype C	(13055)	UK, subtype B	(24422)
	UK, subtypes B & C	(37477)	Africa, all subtypes	(2284)

Table 6.2.: **All training and testing datasets used during this study.** The number of sequences in each dataset is shown in parentheses

should increase the robustness of our classifiers and the relevance of the mutations selected as potentially associated to resistance.

To avoid phylogenetic confounding factors (e.g. transmitted mutations within a specific country or region), and avoid finding mutations potentially specific to a given subtype, we split the training and testing sets by HIV-1 M subtype. This resulted in training a set of classifiers on all subtype B sequences of the UK dataset and testing them on subtype C sequences from the UK dataset, training another set of classifiers on the subtype C sequences of the UK dataset and testing on the subtype B sequences from the UK dataset, as well as training a final set of classifiers on the whole UK dataset, but testing it on the smaller African dataset with a completely different phylogenetic makeup and treatment context.[615] Furthermore, in order to identify novel RAMs and study the behavior of the classifiers, we repeated this training scheme on both datasets, each time removing resistance-associated signal incrementally: first by removing all representation features corresponding to known RAMs from the dataset, and second by removing all sequences that had at least one known RAM. This resulted in each type of classifier being trained and tested 9 times, on radically different sets to ensure the interpretability and robustness of the results (see Table 6.2).

6.2.3. Measuring classifier performance

To compare the performance of our classifiers we used balanced accuracy,[380] which is the average of accuracies (i.e. percentages of well-classified sequences) computed separately on each class of the test set. This score takes into account, and corrects for, the imbalance between RTI-naive and RTI-experienced samples, which would

lead to a classifier always predicting a sequence as RTI-naive getting a classical accuracy score of up to 77% (i.e. the frequency of naive sequences in the UK dataset). We also computed the adjusted mutual information (AMI) between predicted and true sequence labels, which is a normalized version of MI allowing comparison of performance on differently sized test sets.[150] Additionally, mutual information (MI) was used to compute p-values and assess the significance of the classifiers' predictive power. The probabilistic performance of the classifiers was evaluated using an adapted Brier score[660] more suited to binary classification, which is the mean squared difference between the actual class (coded by 1 and 0 for the RTI-experienced and RTI-naive samples respectively) and the predicted probability of being RTI-experienced. This approach refines the standard accuracy measure by rewarding methods that well approximate the true status of the sample (eg. predicting a probability of 0.9 while the true status is 1); conversly, binary methods (predicting 0 or 1, but no probabilities) will be penalized if they are often wrong. The Brier approach thus assigns better scores to methods that recognize their ignorance than to methods producing random predictions.

6.3. Results

6.3.1. Classifier performance & interpretation

As can be seen in Fig 6.1A and 6.1B, when all RAM features and sequences were kept in the training and testing sets, classifiers had good prediction accuracy, with the machine learning classifiers slightly outperforming the "Fisher" classifier. When removing RAM features from the training and testing sets, the classifiers retained a significant prediction accuracy, especially with the African data set and its multiple RAMs that are observed in a large number of sequences (but removed in this experiment). In this configuration the ML classifiers had a similar performance to the "Fisher" classifier, except for the random forest that is slightly less accurate, likely due to overfitting. Also, when removing sequences that had known RAMs, every classifier lost all prediction accuracy, and none could distinguish RTI-naive from RTI-experienced sequences. Regarding the Brier sore, we see the advantage of the machine learning classifiers over the "Fisher" classifier, which is worse than random predictions when known RAMs are removed. The ability of machine learning classifiers to quantify the resistance status should be an asset for many applications.

The fact that classifiers retained prediction accuracy after removing known RAM corresponding features suggests that there was some residual, unknown resistance-associated signal in the data. The fact that this same power was non-existent when removing the known RAM-containing sequences from the training and testing sets, indicates that this residual signal was contained in these already mutated sequences. This suggests that the mutations that are found in the RAM removed experiment (see list below) are most likely accessory mutations that accompany known RAMs.

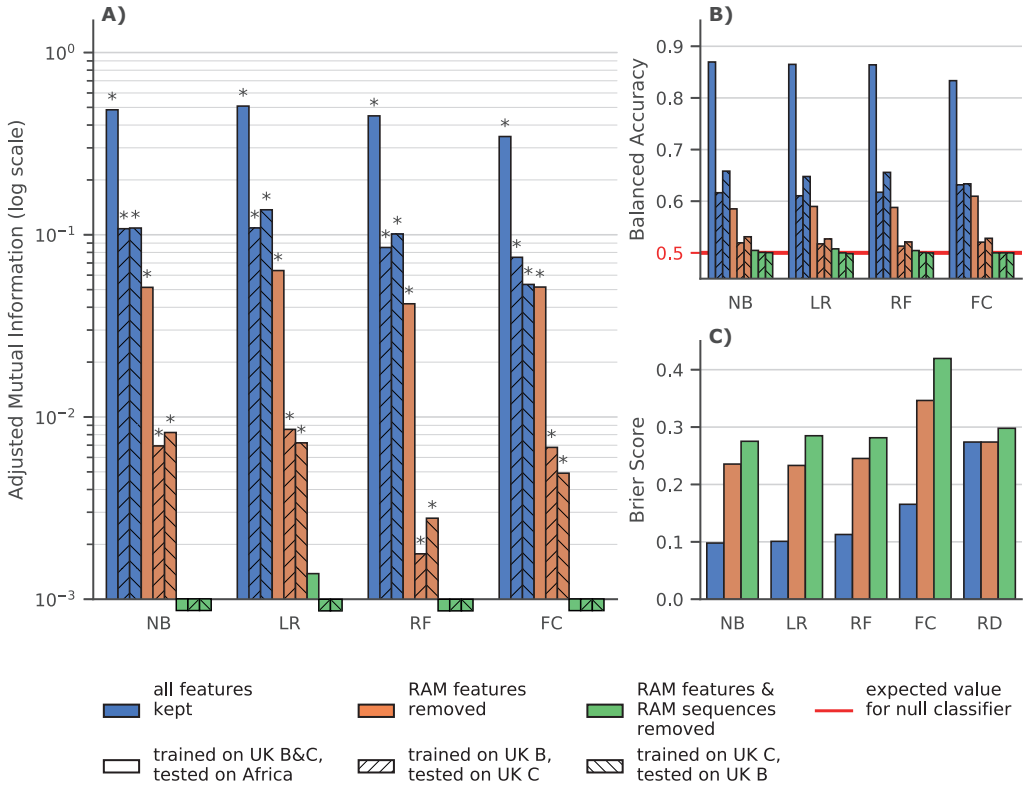


Figure 6.1.: **Classifier Performance on UK and African datasets.** **NB**: naive Bayes, **LR**: Logistic Regression with Lasso regularization, **RF**: Random Forest, **FC**: Fisher Classifier, **RD**: Agnostic random probabilistic classifier (this classifier predicts, as the probability of a sample belonging to a class, the frequency of that class in the training data). **A)** Adjusted mutual information (higher is better) between ground truth and predictions by classifiers trained on dataset with all features (blue), without features corresponding to known RAMs (orange) and without RAM features and without sequences that have at least 1 known RAM (green). Hatching indicates the training set on which a classifier was trained and the testing set on which the performance was measured. The expected value for a null classifier is 0, and 1 for a perfect classifier and a * denotes that the p-value derived from mutual information is ≤ 0.05. For example when trained with all features all the classifiers have a significative MI. Conversly when removing RAM features and RAM sequences none of the classifiers have a significative MI and only LR trained on the entirety of the UK dataset has an AMI $> 10^{-3}$ **B)** Balanced Accuracy score, i.e. average of accuracies per-class (higher is better) for the same classifiers as in a). The red line at $y = 0.5$ is the expected balanced accuracy for a null classifier that only predicts the majority class as well as a random uniform (i.e. 50/50) classifier. **C)** Brier score, which is the mean squared difference between the sample's experience to RTI and the predicted probability of being RTI experienced (lower is better), for the same classifiers as in **A)** and **B)**.

This also suggests that all primary DRMs (i.e., that directly confer antiretroviral resistance) have been identified, which is reassuring from a public health perspective.

The performance discrepancy between the UK and African test sets can be explained by several factors. Firstly, African sequences that have known RAMs are more likely to have multiple RAMs, and thus more (known and unknown) resistance-associated features than their UK counterparts (c.f. Table 6.1). This means that resistant African sequences are easier to detect even when removing known RAMs. Secondly, RTI-naive sequences in the UK test sets are more likely to have known RAMs than their African counterparts (c.f. Table 6.1) and therefore more companion mutations. This means that the RTI-naive sequences in the UK test set are more likely to be misclassified as RTI-experienced than in the African test set.

6.3.2. Additional classification results

The fact that, when looking at classifiers trained without known RAMs , "Fisher" classifiers perform as well as the machine learning ones, leads us to believe that there is little interaction between mutations that would explain resistance better than taking each mutation separately. It is therefore likely that the kind of epistatic phenomena we were looking for, combining several mutations that do not induce any resistance when taken separately, do not come into play here. We are in a classical scheme where primary DRMs confer resistance and associated mutations reinforce the strength of the resistance and/or compensate for the fitness cost induced by primary DRMs.

It is important to remember that in the previous section we were trying (as usual, e.g. see[615]) to find novel mutations associated with resistance by discriminating RTI-naive from RTI-experienced sequences, both with the statistical tests and the classifiers. However, this is intrinsically biased and noisy. Indeed, a RTI-naive sequence is not necessarily susceptible to RTIs as a resistant strain could have been transmitted to the individual. Conversely, an RTI-experienced sequence may not be resistant to treatment, due to poor ART adherence for example. We must therefore keep in mind that the noisy nature of the relationship between resistance and treatment status is partly responsible for the lower performance of classifiers trained on the UK sequences with reduced signal.

Moreover, as all the additional resistance signal we detected is associated to the sequences having at least one known RAM (see above), we performed another analysis trying to discriminate between the sequences having at least one known RAM and those having none. The goal was to check that the mutations we discovered by discriminating RTI-experienced from RTI-naive samples, are truly accessory and compensatory mutations. As can be seen in Fig 6.2A and 6.2B, the classifiers trained to discriminate sequences that have at least one known RAM from those that have

none, on datasets from which all features corresponding to known RAMs were removed, perform much better than classifiers trained to discriminate RTI-experienced from RTI-naive sequences. This increase in performance is especially visible for classifiers tested on UK sequences (more difficult to classify than the African ones, see above), with an AMI often almost one order of magnitude higher for the known-RAM presence/absence classification task. This further reinforces our belief that all there is a fairly strong residual resistance-signal in sequences that contain known RAMs, due to new accessory and compensatory mutations identified by our classifiers and Fisher tests. As a side note, Logistic regression (LR) consistently outperforms other classifiers, a tendency already observed in Fig 6.1.

6.3.3. Identifying new mutations from classifiers

We assessed the importance of each mutation in the learned internal model of all the classifiers, in the setting where all known RAMs have been removed from the training dataset. For the Fisher classifier, we used one minus the p-value of the exact Fisher test as the importance value, therefore the more significantly associated mutations have the higher importance value and were ranked first. For a given classification task, we ranked each mutation according to the appropriate importance value for each classifier (see above), trained on the B or C subtypes, with the highest importance value having a rank of 0. We then computed the average rank for each mutation and each classification task (RTI-naive/RTI-experienced and RAM present/RAM absent). This gave us, for each classification task, a ranking of mutations potentially associated with resistance that took into account the importance given to this new mutation by each classifier trained on this task. Mutations that were in the 10 most important mutations for both of the classification tasks were considered of interest. Based on these criteria we selected the following potentially resistance-associated mutations (w.r.t. the HXB2 reference genome): L228R, L228H, E203K, D218E, I135L and H208Y. These mutations are referred to as "new mutations" in the rest of this study.

To check the epistatic nature of these selected mutations we computed the relative risk $RR(new, X)$ between a new mutation and a binary character X. $RR(new, X)$ was computed from the contingency table between new and X as follows:

	X present	X absent
new present	A	B
new absent	C	D

$$RR(new, X) = \frac{A}{A+C} \div \frac{B}{B+D}$$

The RR gives us a measure for how over-represented each of our new mutations is in sequences that have the X character compared to those that don't.

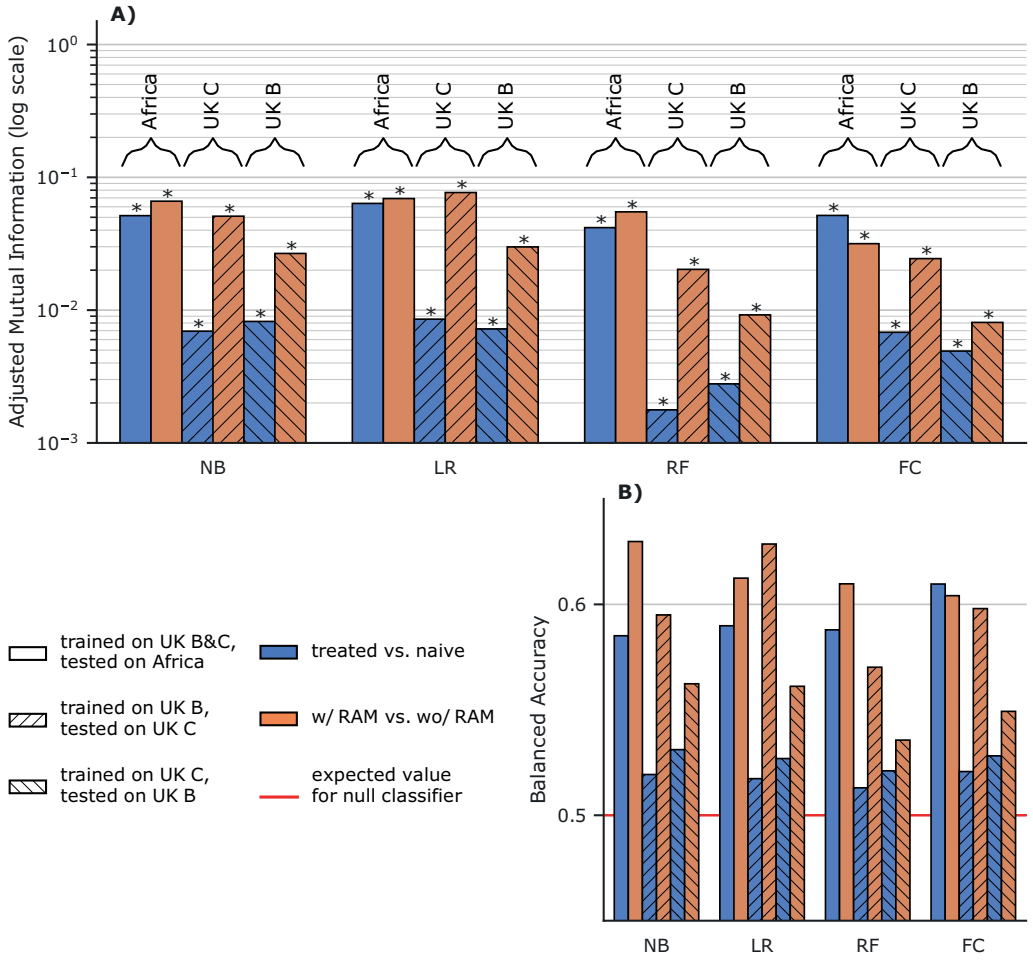


Figure 6.2.: **Discrimination between sequences having at least one RAM, and those having none on sequences with training features corresponding to known RAMs removed.**
NB: naive Bayes, **LR**: Logistic Regression with Lasso regularization, **RF**: Random Forest, **FC**: Fisher Classifier. **A)** Adjusted mutual information (higher is better) for classifiers trained without features corresponding to known RAMs. The classifiers are either trained to discriminate RTI-naive from RTI-experienced sequences (blue), or sequences with at least one known RAM from sequences that have none (orange). Hatching and braced annotations indicate the training and testing sets resulting in a given performance measure. **B)** Balanced accuracy, i.e. average of accuracies per-class for the same classifiers as in **A)** (higher is better). The red line at $y = 0.5$ is the expected value for a classifier only predicting the majority class as well as a random uniform (50/50) classifier.

To get a general idea of this over-representation, for each new mutation we computed $RR(new, treatment)$ comparing the prevalence of the new mutation in RTI-experienced and RTI-naive sequences. We also computed $RR(new, withRAM)$ comparing the prevalence the new mutation in sequences having at least one known RAM and sequences that have none. Both of these RRs are shown in Table 6.3 for each new mutation.

We then computed $RR(new, RAM)$ for each known RAM present in more than 0.1% of UK sequences and the new mutations. In Fig 6.3 we see the RRs for which the lower bound of the 95% confidence interval, computed on 1000 bootstrap samples from the UK dataset, was greater than 4.

6.3.4. Detailed analysis of potentially resistance-associated mutations

As can be seen in Table 6.3, all of these new mutations except for I135L, are highly over-represented in RTI-experienced sequences and sequences that already have known RAMs, with lower bounds on the 95% RR CI always greater than 5, and often exceeding 10. When looking at the RRs computed for individual RAMs on the UK dataset (Fig 6.3), this impression is confirmed with very high over-representation of these new mutations potentially associated with resistance in sequences that have a given known RAM, with 95% RR lower CI bounds sometimes greater than 80 (H208Y/L210W and D218E/D67N), and most of the time greater than 10. with the noticeable exception of I135L where only 2 known RAMs give RRs with lower CI bounds greater than 4. The RRs computed on the African dataset (C.1) tell a similar story albeit with smaller RR values due to a smaller number of occurrences of both new mutations and known RAMs.

The genetic barrier to resistance for each of these new mutations is quite low, with a minimum of 1 base change for each of them (Table 6.3). We also computed the average codon distance (i.e. number of different bases), weighted by the prevalence of wild and mutated codons at the given positions in the UK (Table 6.3) and Africa (Table C.5) datasets, and in each case the average codon distance was always close to 1. In other words, at the amino acid level these mutations are expected to be relatively frequent. However, their frequencies are much higher in treated/with-RAM sequences than in naive/without-RAM ones (Table 6.3). Moreover, if we look at the BLOSUM62 scores (Table 6.3), some of these mutations induce some substantial changes in physicochemical properties, most notably at site 228, which reinforces again the likelihood that these mutations are associated with resistance. These metrics were also computed for all known RAMs (Table 6.3). For all these metrics, and the 6 new potential RAMs, values are contained between the 5^{th} and 95^{th} percentiles computed on known RAMs, except for the BLOSUM score of L228H that corresponds to a drastic physicochemical change.
To gain more insight on these new mutations we also observed their spatial location on the 3-D HIV-1 RT structure using PyMol.[666] HIV-1 RT is a heterodimer with two

	codon distance			UK			
					$RR(new, X)$		
	min	avg	B62	count	*treatment*	*any RAM*	p-value
L228R	1	1.16	-2	227 (0.4%)	18.1 [12.9;27.3]	115.7 [55.1;507.3]	$3.4 \cdot 10^{-31}$
E203K	1	1.31	1	256 (0.5%)	11 [8.2;15.1]	20.1 [13.7;32.1]	$1.1 \cdot 10^{-14}$
D218E	1	1	2	168 (0.3%)	13.1 [9.0;19.6]	27 [16.3;57.0]	$3.3 \cdot 10^{-10}$
L228H	1	1.12	-3	287 (0.5%)	6.4 [5.1;8.4]	9.2 [6.9;12.6]	$4.4 \cdot 10^{-16}$
I135L	1	1.16	2	540 (1.0%)	1.8 [1.5;2.1]	2.4 [2.0;2.8]	$5.9 \cdot 10^{-08}$
H208Y	1	1.10	2	205 (0.4%)	8.8 [6.5;12.5]	14.9 [9.9;23.6]	$1.2 \cdot 10^{-05}$
RAMs	1 [1;2]	1.35 [1;2.44]	0 [-2;3]	58 (0.1%) [2;1842]	8.3 [0.6;∞]	26.4 [1.4;∞]	$3.1 \cdot 10^{-2}$ [$2.3 \cdot 10^{-58}$;1]

Table 6.3.: **Analysis of new potential RAMs.**
Codon distance: For each new mutation we computed the minimum number of nucleotide mutations to go from the wild amino acid codons to those of the mutated amino acid, as well as the average codon distance between both amino acids, weighted by the prevalence of each wild and mutated codon at the given position in the UK dataset. **B62:** BLOSUM62 similarity values (e.g. D218E = 2, reflecting that E and D are both negatively charged and highly similar). **Count:** We looked at the number of occurrences of each new potential RAM in the UK dataset and the corresponding prevalence in parentheses. **Relative risks:** We computed $RR(new, treatment)$ (e.g. L228R is 18.1 times more prevalent in RTI-experienced sequences compared to RTI-naive sequences in the UK dataset). We also computed $RR(new, any\ RAM)$ (e.g. L228R is 115.7 times more prevalent in sequences that have at least one known RAM than in sequences that have none in the UK dataset). The 95% confidence intervals shown under each RR were computed with 1000 bootstrap samples of size $n = 55,000$ drawn with replacement from the whole UK dataset. **p-values:** Fisher exact tests were done on the African dataset (to avoid confounding effects due to phylogenetic correlation) to see if each of these new mutations were more prevalent in RTI-experienced sequences. The same metrics were computed for all known RAMs, the median values are shown in the last two lines of this table, as well as the 5^{th} and 95^{th} percentiles which are shown underneath. $RR(RAM, any\ RAM)$ values were computed for any RAM except itself to avoid always having infinite ratios.

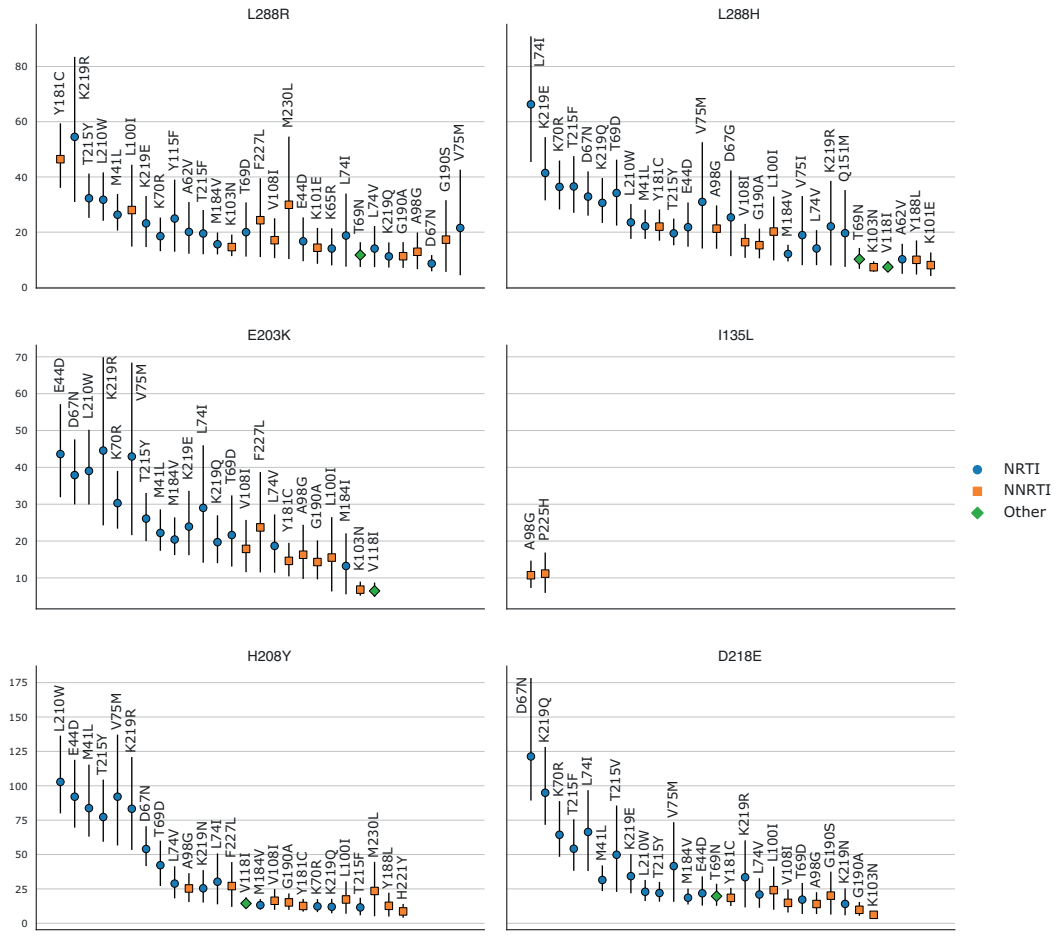


Figure 6.3.: **Relative risk of the new mutations with regards to known RAMs on the UK dataset.**
(i.e. the prevalence of the new mutation in sequences with a given known RAM divided by the prevalence of the new mutation in sequences without this RAM). RRs were only computed for mutations (new and RAMs) that appeared in at least 0.1% (=55) sequences. 95% confidence intervals, represented by vertical bars, were computed with 1000 bootstrap samples of UK sequences. Only RRs with a lower CI boundary greater than 4 are shown. The shape and color of the point represents the type of RAM as defined by Stanford's HIVDB. Blue circle: NRTI, orange square: NNRTI, green diamond: Other. RR values are shown from left to right, by order of decreasing values on the lower bound of the 95% CI.

subunits translated from the same sequence with different lengths and 3-D structures. The smaller p51 subunit (440 AAs) has a mainly structural role, while the larger p66 (560 AAs) subunit has the active site at positions 110, 185 and 186. The p66 subunit also has a regulatory pocket behind the active site: the non-nucleoside inhibitor binding pocket (NNIBP) formed of several sites of the p66 subunit as well as site 138 of the p51 subunit. Nucleoside RT Inhibitors (NRTI) are nucleotide analogs and bind in the active site, blocking reverse transcription. Non-Nucleoside RT Inhibitors (NNRTI) bind in the NNIBP, changing the protein conformation and blocking reverse transcription. More details on the structure and function of HIV-1 RT can be found in.[564] A general view of where the new mutations are situated with regards to the other important sites of HIV-1 RT is shown in Fig 6.4, and is detailed below.

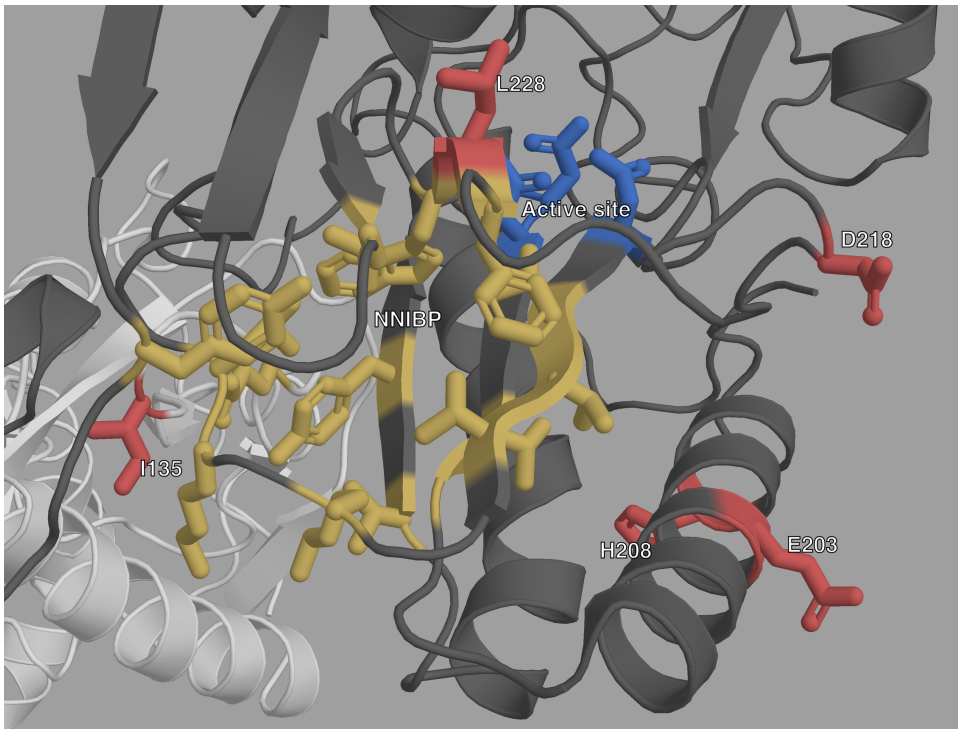


Figure 6.4.: **Structure of HIV-1 RT with highlighted important sites.** The p66 subunit is colored dark gray and the p51 subunit white. The active site is highlighted in blue, and the NNIBP is highlighted in yellow. The sites of new mutations are colored in red.

6.3.4.1. L228R / L228H

L228R is the most important of these new mutations according to the feature importance ranking done above. This is reflected in the very high over-representation

in RTI-experienced sequences and sequences with known RAMs shown in Table 6.3 . When looking at the detailed RRs shown in Fig 6.3, we observe that L228R presents high RR values with mainly NRTI RAMs, but also with NNRTI RAMs such as Y181C and L100I, and this is even more so h for RRs computed on the African dataset (C.1). L228H is very similar in all regards to L228R, however its highest RRs are exclusively with NRTI RAMs.

Site 228 of the p66 subunit is located very close to the active site of RT, where NRTIs operate (Figs 6.4 and C.3) which could explain the role that L228R and L228H seem to have in NRTI resistance. However, site 228 of the p66 subunit is also between sites 227 and 229 which are both part of the NNIBP. Furthermore, both L228H and L228R have very low BLOSUM62 score, of -3 and -2 respectively (Table 6.3). Arginine (R) and Histidine (H) are both less hydrophobic that Leucine (L), and have positively charged side-chains. This important change in physicochemical properties could explain the role they both seem to have in NRTI resistance. However, while both Arginine and Histidine are larger than Leucine, Arginine is also fairly larger than Histidine, which is aromatic. This difference between both residues might explain the association L228R seems to have with NNRTI resistance that L228H does not have.

6.3.4.2. E203K / H208Y

Both E203K and H208Y are highly over-represented in RTI-experienced sequences and sequences with known RAMs. They both have high RR values for NRTI RAMs. Furthermore the most highly valued RAM RRs in Fig 6.3, are very similar for E203K and H208Y. Structurally they are close to each other on an alpha helix which is close to the active site.

Both E203K and H208Y have positive, albeit not maximal, BLOSUM62 scores, meaning they are fairly common substitutions. However, these mutations induce some change in physicochemical properties with Tyrosine (Y) being less polar than Histidine (H), and the change from Glutamic Acid (E) to Lysine (K) corresponding to a change from a negatively charged side chain to a positively charged one.

All this, combined with their structural proximity and the shared high RR values for single RAMs, suggests a similar role in NRTI resistance.

6.3.4.3. I135L

In Table 6.3 and Fig 6.3, we observe that I135L has the lowest RR values of all the new mutations, with CI bounds lower than 2 in Table 6.3's general RRs. However, it is the most prevalent of the new mutations. If we look at the detailed RRs of Fig 6.3, we see that I135L is significantly over-represented in sequences with NNRTI RAMs, specifically A98G and P225H. Structurally this makes sense: On the p66 subunit,

site 135 is on the outside, far from both the active site and the NNIBP. However, site 135 on the p51 subunit is located very close to the NNIBP (Figs 6.3 and C.2).

The BLOSUM62 score for this substitution is quite high (Table 6.3), which is expected since both residues are very similar to one another, differing only by the positioning of one methyl group. However, Leucine (L) is less hydrophobic than Isoleucine (I), despite they are still both classified as hydrophobic residues (Table C.5).

The proximity between site 135 and the pocket in which NNRTI RAMs bind, as well as the high RR values for these NNRTI RAMs leads us to believe that I135L could play a subtle accessory role in NNRTI resistance, either by enhancing the effect of some NNRTI RAMs (typically, A98G and P225H), or by compensating for loss of fitness.

6.3.4.4. D218E

D218E is also highly over-represented in both RTI-experienced sequences and sequences with known RAMs. It has infinite RR values in the African dataset (Table 6.3), because it is quite rare in this dataset, and all of its 25 occurrences are in sequences that have at least one known RAM and are RTI-experienced. In fact, from the UK dataset we can see that D218E has some of the highest RR values for individual RAMs (along with H208Y). The majority of these very high RR values occur for NRTI RAMs. Site 218 on the p66 subunit is quite close to the RT active site, which could explain the role D218E seems to have in NRTI resistance. Aspartic acid (D) and Glutamic acid (E) are very similar amino acids, both acidic with negatively charged side-chains, as reflected in their fairly high BLOSUM62 score, the main difference between both being molecular weight, with E being slightly larger than D.

6.4. Discussion and perspectives

Our method has allowed us to identify six mutations that might play a role in drug resistance in HIV. These mutations are significantly over-represented in RTI-experienced sequences, as well as sequences exhibiting at least one other known RAM. The fact that models trained on the UK are still performant on such a different dataset as the African one strongly suggests that the learned classifier models have acquired generalized knowledge on resistance. For all of these new mutations their spatial positioning on HIV-1 RT is consistent with our conclusions, as all were either close to the active site or the regulatory binding pocket.

Some of the mutations we have identified as potentially associated with resistance have been mentioned in previous studies. L228R/H have been observed before[667]

and were suggested to be associated with reduced susceptibility to didanosine.[668,669] I135L has been observed in sequences with reduced susceptibility to NNRTIs.[670] H208Y has been associated with NNRTI and NRTI resistance[671] and it has been suggested that it has an accessory role in NRTI resistance.[672] E203K, D218E, L228RH and H208Y have all been mentioned in[673] as probably linked to phenotypic resistance to NRTI and NNRTI.

However, none of these mutations has been experimentally confirmed as conferring or helping with drug resistance to the best of our knowledge. The fact that we find them again with a big data analysis of highly different sequences and involved statistical selection procedure combining multiple testing and machine learning, and that we have very high significance, clearly indicates their potential role in resistance. Therefore, we believe they are sufficiently linked to drug resistance that they garner a closer inspection either in-vitro or in-vivo to determine the mechanisms that could allow them to play a role in resistance.

With our machine classifiers we seem to have found some RAMs of an accessory nature, over-represented in sequences already containing known RAMs. This is a form of epistasis, where the interaction between the main RAM and the accessory RAM is important. However, we did not manage to find subtler forms of epistasis, in our dataset, where two mutations separately have no effect on resistance but have an effect together. This is partly indicated by the fact that there is a limited performance gap between the Fisher exact tests and more sophisticated classifiers, that are able to reveal significant association of mutations, while each individual mutation has low prediction power. However, one advantage of machine learning classifiers, is that they are probabilistic, meaning that they can give more nuanced insights into the nature or resistance level of a given sequence than the classical binary presence/absence of RAMs approach. In this regard logistic regression appears as a method of choice, showing similar or better performance than other classifiers, and an easy interpretation that is facilitated by the lasso regularization which performs a simple feature selection and retains the most important ones. Similar results were already observed on other sequence analysis tasks.[674] In order to investigate the second form of epistasis further we tested each pair of mutations in the UK dataset ($n = 867,903$) with Fisher exact tests to see if they were linked to treatment status. In order to mitigate the effects of phylogenetic correlation which are sure to have an effect in this type of setting, we tested the pairs that were significantly associated to treatment ($n = 1,309$) again on the African dataset. We also compared these results to the Fisher exact tests executed for each single mutation. We did not find any pair of mutations that was significantly associated, to treatment where neither member were significantly associated individually. Moreover, we only found 3 significantly associated pairs of mutations that did not include at least one known RAM, and they all included one of our newly found potential RAM: L228R + I142V, L228R + F214L and L228H + F214L (see appendix C.6 for details).

With therapeutic strategies targeting multiple proteins that are now used, there

might be some epistatic effects with other regions of the HIV genome that are targeted by some of the drugs. These potential effects however, lie outside the scope of this study.

Because of the lack of detailed treatment history metadata, we did not distinguish mutations arising from NRTIs or NNRTIs. We believe that a large amount of high quality sequence data, along with a sufficiently detailed log of treatments and drugs the sequences were exposed to, could allow us to use our machine-learning approach to find mutations related to specific drugs and thus furthering our knowledge of HIV drug resistance, giving clinicians more tools to manage and help infected patients.

Acknowledgments

We thank Anna Zhukova, Frédéric Lemoine and Marie Morel for their help and suggestions.

We also thank the UK HIV Drug Resistance Database and the UK Collaborative HIV Cohort:

Steering committee: David Asboe, Anton Pozniak (Chelsea & Westminster Hospital, London); Patricia Cane (Public Health England, Porton Down); David Chadwick (South Tees Hospitals NHS Trust, Middlesbrough); Duncan Churchill (Brighton and Sussex University Hospitals NHS Trust); Simon Collins (HIV i-Base, London); Valerie Delpech (National Infection Service, Public Health England); Samuel Douthwaite (Guy's and St. Thomas' NHS Foundation Trust, London); David Dunn, Kholoud Porter, Anna Tostevin, Oliver Stirrup (Institute for Global Health, UCL); Christophe Fraser (University of Oxford); Anna Maria Geretti (Institute of Infection and Global Health, University of Liverpool); Rory Gunson (Gartnavel General Hospital, Glasgow); Antony Hale (Leeds Teaching Hospitals NHS Trust); Stéphane Hué (London School of Hygiene and Tropical Medicine); Michael Kidd (Public Health England, Birmingham Heartlands Hospital); Linda Lazarus (Expert Advisory Group on AIDS Secretariat, Public Health England); Andrew Leigh-Brown (University of Edinburgh); Tamyo Mbisa (National Infection Service, Public Health England); Nicola Mackie (Imperial NHS Trust, London); Chloe Orkin (Barts Health NHS Trust, London); Eleni Nastouli, Deenan Pillay, Andrew Phillips, Caroline Sabin (University College London, London); Kate Templeton (Royal Infirmary of Edinburgh); Peter Tilston (Manchester Royal Infirmary); Erik Volz (Imperial College London, London); Ian Williams (Mortimer Market Centre, London); Hongyi Zhang (Addenbrooke's Hospital, Cambridge).

Coordinating Center: Institute for Global Health, UCL (David Dunn, Keith Fairbrother, Anna Tostevin, Oliver Stirrup)

Centers contributing data: Clinical Microbiology and Public Health Laboratory, Addenbrooke's Hospital, Cambridge (Justine Dawkins); Guy's and St Thomas' NHS

Foundation Trust, London (Emma Cunningham, Jane Mullen); PHE – Public Health Laboratory, Birmingham Heartlands Hospital, Birmingham (Michael Kidd); Antiviral Unit, National Infection Service, Public Health England, London (Tamyo Mbisa); Imperial College Health NHS Trust, London (Alison Cox); King's College Hospital, London (Richard Tandy); Medical Microbiology Laboratory, Leeds Teaching Hospitals NHS Trust (Tracy Fawcett); Specialist Virology Centre, Liverpool (Elaine O'Toole); Department of Clinical Virology, Manchester Royal Infirmary, Manchester (Peter Tilston); Department of Virology, Royal Free Hospital, London (Clare Booth, Ana Garcia-Diaz); Edinburgh Specialist Virology Centre, Royal Infirmary of Edinburgh (Lynne Renwick); Department of Infection & Tropical Medicine, Royal Victoria Infirmary, Newcastle (Matthias L Schmid, Brendan Payne); South Tees Hospitals NHS Trust, Middlesbrough (David Chadwick); Department of Virology, Barts Health NHS Trust, London (Mark Hopkins); Molecular Diagnostic Unit, Imperial College, London (Simon Dustan); University College London Hospitals (Stuart Kirk); West of Scotland Specialist Virology Laboratory, Gartnavel, Glasgow (Rory Gunson, Amanda Bradley-Stewart).

Supporting information

Supporting Information can be found in the appendix C

7. Learning Alignments, an Interesting Perspective

Recently, machine learning methods have been increasingly applied to the process of alignment. Using this framework to "learn" an optimal sequence alignment algorithm might lead to better performance with less design biases.

7.1. Deep learning and sequences

As many of these techniques are based on deep learning, which I did not introduce in Chapter 4, I will first present deep learning very shortly. I will then introduce the concept of learned sequence embeddings which could become very useful for machine sequence alignment.

7.1.1. Intro to deep learning

Deep learning is the process of learning using neural networks. Neural networks all started in 1958 when Rosenblatt proposed the *perceptron*.[675] This learning algorithm was loosely inspired by biological neurons, which led to the name: *neural networks*. The perceptron takes as input n values, these are used in a weighted sum that is then fed through an *activation function*. The output of this function is the output of the perceptron. Originally, to replicate biological neurons, the activation function was a step function where, the perceptron has an output only if the weighted sum crosses a given threshold. This structure is often represented through a computational graph as in Figure 7.1. By tuning the weights of the inputs, the perceptron can be used to solve linear separation problems.

The perceptron could only be used on simple linear separation problems, but it was discovered that by linking several perceptrons together, mimicking a biological brain, more complex problems could also be solved. These structure, called *multilayer perceptrons* (MLP), are organized in layers where the outputs of perceptrons on a layer are used as inputs by perceptrons is the next layer (c.f. Figure 7.2). The perceptrons, when in this form, are often called *neurons*, and the MLP a *neural network* (NN). These neural networks are organized in layers, with an input and output layer on either end, and hidden layers in the middle. With the large number

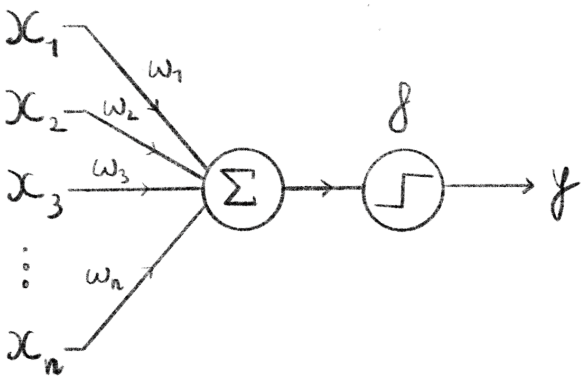


Figure 7.1.: **Computational graph of a perceptron.**
Here, n inputs $\{x_1, \dots, x_n\}$ are passed into the perceptron where they are summed, weighted by $w_1, \dots, w_n$. This sum is then fed through the perceptron's activation function f (here a binary step function) which gives the output y of the perceptron. Often the sum is implicitely considered part of the activation function, and will be represented as a single node in computational graphs.

of weights to tune, these models were very difficult to train and therefore not very practically useful.

There was a great resurgence of these models in the nineties due to the invention of *backpropagation*.[676] By replacing the step functions of neurons with continuous, differentiable activation functions like sigmoids or hyperbolic tangents, a gradient of the output could be computed w.r.t each weight. This made gradient descent procedures possible for automatically learning the optimal weights from data as (c.f. Section 4.1.1). With this method, neural networks could be efficiently trained on complex classification and regression problems.[677] It was also proven that with hidden layers, neural networks are universal function approximators,[678–680] suitable for all types of tasks. One notable caveat for neural networks is, due to the large amount of weights to tune, that they require large amounts of training data, which also explains their low usage before the internet and accompanying data explosion.

In the following years, NNs saw an large increase in usage, with more complex architectures like convolutional neural networks (CNN) achieving state of the art results in computer vision tasks.[681,682] By representing an input variable as a linear combination of its neighbors, some form of contextual information can be passed to the NN and improve performance. CNNs can also have good results in non computer-vision tasks like: drug resistance prediction,[331] protein subcellular localization,[352] or epidemiological model parameter estimation.[683]

More recently, as computational power and the amount of training data grew, larger and deeper (i.e. more hidden layers) architectures were able to be trained

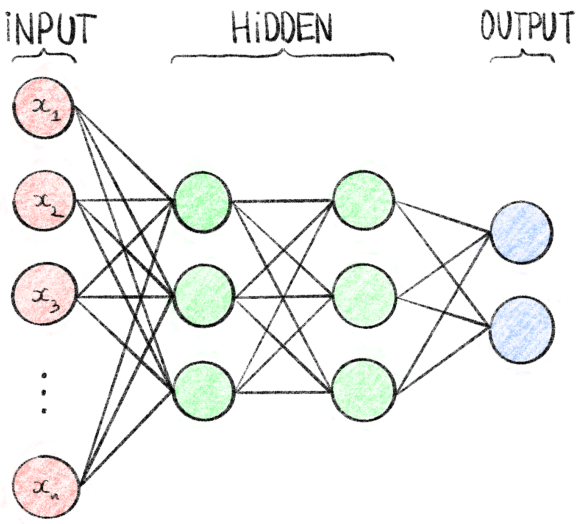


Figure 7.2.: **Computational graph of a multilayer perceptron.**
This MLP, also called feedforward neural network, has n inputs $\{x_1, \dots, x_n\}$ represented as the input layer, 2 hidden layers of 3 neurons each and an output layer of 2 neurons (e.g. suitable for binary classification). It is fully connected meaning that each node of a given layer is used as input for every neuron of the following layer. Each edge in this graph corresponds to a weight which are the tunable parameters during the training process.

and achieved state of the art performance in many fields: image recognition with deep CNNs like `Alexnet`[684] or `Resnet`,[685] translation with Recurrent NNs[686] and Transformers[687] (more on that in Section 7.1.2) or protein structure prediction with `Alphafold2`.[137]

7.1.2. Learned sequence embeddings

An area that in which deep learning has recently proved particularly useful, is the creation of relevant learned embeddings. These embeddings, similarly to the encodings discussed in Section 4.3, transform a sequence of categorical tokens in a numerical vector which can then be used in machine learning tasks. By learning these embeddings, the hope is that the resulting vector will retain the most important information in the sequence and some contextual information.

7.1.2.1. x-`2vec`

Learned embeddings were mainly developed in the field of natural language processing (NLP), where machine learning algorithms use text, in languages such as English or French, as input. In this contexts, simple encodings like OHE are not very practical because of the very high dimensionality of a language. For example, the Merriam-Webster English dictionary contains 470,000 words[688] so to encode a single word with OHE would result in a 470,000-dimensional sparse vector. Encoding a whole text or even a single sentence is wildly unpractical. Therefore, as a field, NLP needed to come up with ways of efficiently representing words in lower-dimensional vectors than naive encoding methods, while retaining semantic meaning.

One of the early methods for creating such embeddings is called `word2vec`,[689,690] proposed by researchers at Google, that learns a word-embedding on a particular text corpus. This method is designed to make embeddings that contain semantically relevant information. An example given in the article is that the vector corresponding to $vec(Madrid) - vec(Spain) + vec(France)$ should be very similar to the vector $vec(Paris)$, and that similar words should result in similar vectors.

The way this method works is by considering a word within its context, *i.e.* a window of length k centered around the word. In a corpus of words (*i.e.* our training data), each word is encoded as a One Hot Vector, which is possible since the corpus contains only a subset of the words in the English language. A neural network is then trained on one of two tasks:[691]

- Continuous bag of words: where the word is predicted given the context of the word as input
- Skip-gram: where the context is predicted given the encoded word vector

After having sufficiently trained the neural network on the corpus on one of these tasks, one of the hidden layers of the network can be extracted and used as a vector representation of the input word, this results in an embedding method that is specific to a given corpus and the embedded vectors can be used in downstream learning tasks.

`word2vec` was very successful and widely used in the field of NLP, it is perhaps no surprise that the ideas behind it were adapted and reused in the field of bioinformatics. `dna2vec`[692] uses similar ideas and was used to embed k-mers, and predict methylation sites on DNA sequences.[693] Similar embedding methods like `seq2vec`[694] as well as `bioVec` (including the protein specific `protVec`)[695] were also developed to embed whole biological sequences. They were successfully used in biological sequence classification problems.[696]

7.1.2.2. The attention revolution

While `word2vec` was widely used for many NLP tasks where word embeddings were needed, a lot of interesting developments on word embeddings were made in the field of automated machine translation. In this application, the desired embedding characteristics are slightly different. While semantic relevance is useful, in machine translation the embedding method needs to be able to capture dependencies, e.g. within a sentence where the link between the subject and the verb must be captured even though they are not necessarily next to each other. This was initially done by using recurrent neural networks, called RNNs or LSTMs, but they were hard to train and had trouble properly capturing long-range dependencies.[697]

One of the most successful methods developed for this task is the transformer,[687] also created by Google researchers. The main mechanisms of the transformer is the *self-attention* mechanisms: each input token, usually encoded as a One-Hot vector, is represented as a weighted sum of all the other tokens in a sequence *(here a token is a word and the sequence is a sentence)*. The weights of this sum are trained along with the rest of this network. By stacking several of these self-attention blocks, transformers can learn to represent and leverage long-range dependencies. These transformers are made of an encoder module that learns the token embedding, and a decoder module that makes predictions when fed embedded tokens sequentially. This mechanism, attention, and the transformer in general have had very successful applications in machine translation, while being easier to train than recurrent networks.[698]

This architecture was used to create very large pre-trained language models, that is to say models that perform word embedding. These models like `BERT`[699] or `GPT-3`[700] are huge, with millions or even billions of learned weights, and have been trained on huge quantities of data in order to produce word embeddings useful in a wide range of contexts. `BERT` was trained using Masked Language Modelling (MLM), where

some percentage of the tokens *(words)* in an input sequence *(sentence)* are replaced by a special `[MASK]` token, and the model is trained to predict the whole sentence, effectively guessing what words are missing based on the context of the whole sequence. This process allows the model to learn relevant dependencies between tokens in the training data.

As was the case with `word2vec`, these methods have been adapted to bioinformatics tasks with state of the art results, proving the versatility of the transformer model. Several *protein language models* similar to `BERT` were trained on various training sets of protein data like `ProGen`,[701] `ProGen2`[702] and `ProtBERT`.[373] These large protein language models have been studied and interesting properties have been observed.[703] Some specific characteristics of proteins can be inferred from these models without specifying them in the training step. For example, protein language models seem to learn some information about the protein structure and attention maps can be used to infer residue contact maps.[704–706] Similarly these models capture some information about protein function,[707] mutational effects,[708] evolutionary characteristics[709] and can even be used to generate new protein with desired properties.[701] Some large language models have also been trained on DNA sequences like `DNABert`[710] and also seem to capture some information without explicit specification during training, like variant effects.[711]

While, these protein language models have shown very useful for embedding single sequences, some developments have been made to embed multiple sequence alignments as learning inputs. In some cases this is done by including information on the alignment in the tokens and then using a regular language model to embed them.[712] In the case of the MSA transformer,[713] the attention mechanism was extended to include a weighted sum between aligned sequences effectively taking the alignment into account when embedding sequences. An attention-like mechanism was also used to train a protein structure prediction model directly on MSAs.[714] Similarly, by pre-training language models on profiles derived from MSAs, some information about the alignment can be included in the resulting embeddings.[715] Finally aligned sequences can be used as inputs in a regular transformer as was done `DeepConsensus`,[716] a transformer-based polisher to decrease the error rate PacBio HiFi reads even further. Finally the *EvoFormer* model included in `AlphaFold2`,[137] which embeds MSAs to predict protein structure, is partly responsible for the leap in performance between the two generations of the `AlphaFold` model, and the current protein structure prediction revolution.[717]

It is important to note that while these transformer models are very powerful and useful in practice, their complexity and size makes them very hard to study and understand what they actually learn. There is work to peek inside this "black box", notably by interpreting the learned attention maps[718] and decipher biologically relevant information contained within.

7.2. Learning sequence alignment

With the success of deep learning methods in learning informative and effective embeddings from sequences, it is maybe natural to try and see if similar methods can learn how to align sequences to each other.

7.2.1. Predicting a substitution matrix

One approach is to learn a *position-specific scoring matrix* (PSSM), which assigns an alignment cost not between two amino-acids but between two specific residues of the sequences (i.e. an amino acid/position pair). Therefore, when aligning a sequence of length m and another of length n, we can use a standard alignment method such as NW or SW with an $m \times n$ PSSM.

One approach, used in the `SAdLSA` model[719] used CNNs to refine an input PSSM. The model is trained on experimentally validated structural alignments. A starting PSSM is created from both sequences with `PSI-BLAST`,[200] and fed through a deep CNN, which outputs a refined PSSM. This learned matrix is used with a SW algorithm to locally align the two sequences. This alignment is then compared to the structural alignment to compute a loss and train the model.

Some methods rely on protein language model embeddings coupled with differentiable alignment algorithms to learn a PSSM in an end-to-end fashion. `DeepBLAST` is one such model.[720] It was trained on 1.5 million structural alignments obtained from the PDB database.[721] The sequences are embedded using a pre-trained LSTM-based protein language model. These embeddings are fed through LSTM networks to predict a match scoring and gap scoring PSSMs. These matrices are then used in a differentiable variant of the NW algorithm, that can be used to backpropagate the alignment error through the network and learn relevant parameters. RNNs and LSTMs were also used to predict PSSMs by Guo *et al.* albeit with the goal of protein structure prediction rather than alignment.[722]

The `DEDAL` model[723] implements similar ideas. It predicts matching, gap-open and gap-extend PSSMs from a pair of sequences, that can be used in a classical alignment method, in this case a SW algorithm. In this model, a transformer-based embedding network is used to embed each residue of both sequences. Then each possible pair of embedded residues from both sequences is used to predict specific gap-open, gap-extend and match scores used to build the PSSMs. The `DEDAL` model is trained on three tasks at once:

1. Masked language modelling (c.f. Section 7.1.2.2) to train the transformer-based embedding model on 30 million sequences from the UniRef50 database.[724]

2. A homology detection task where the whole model is trained to predict if a pair of sequences are evolutionarily related or not. This was done on pairs of sequences extracted from the 1.2 million sequences of the Pfam-A seed database.[725]
3. An alignment task, where the whole model is trained to align two sequences using the authors' differentiable variant of the SW algorithm to backpropagate the alignment error through the network and tune the parameters. This training task was also done using aligned sequence pairs from the Pfam-A seed database.

Trained on the three tasks at once, the `DEDAL` model predicts PSSMs leading to good alignments overall. However, where it really shines and outperforms other methods is on alignments of remote homologs. Classical alignment algorithms can struggle when the similarity between two sequences dips below a certain threshold, `DEDAL` is able to pick up on this remote homology and produce a sensible and accurate alignment.

The learned alignment module[726] also uses a differentiable variant of the SW algorithm to learn a scoring matrix. Sequences are encoded as OHE vectors and fed embedded with simple convolutions, to predict a "context-specific" scoring matrix. This module is used to build MSAs where, similarly to the center star alignment, all *target* sequences are aligned to a single *query* sequence. This model was validated by including it in the `Alphafold2` model and seeing the improvement in performance for certain protein structure prediction tasks.

7.2.2. Predicting an alignment

Predicting a PSSM is one way of learning to align. However, an alignment algorithm still needs to be used in order to obtain aligned sequences. It might be possible to directly output an alignment between input sequences. As stated above, transformers have been particularly useful in automated translation, and one could construe the alignment problem as translating from an unaligned sequence "language" to an aligned sequence "language". This is exactly the idea behind `BetaAlign`, a recently developed transformer model used for pairwise and multiple sequence alignment.[727] For example, the two sequences `AAG` and `ACGG` can be represented as a single "sentence": `AAG|ACGG` with the `|` special token denoting a separation between sequences. Aligned sequences output by the transformer can then be represented as a succession of aligned pairs: `AAAC-GGG` corresponding to the following alignment:

```
AA-G
ACGG
```

The authors trained this model on millions of simulated alignments, of two to ten sequences, generated with different underlying evolutionary models, in the same

fashion that regular transformers are trained for machine translation. The authors trained models for protein and DNA sequence alignment on these simulated datasets, containing sequences around 40 residues long. According to some measures, `BetaAlign` outperforms widely used multiple sequence aligners such as `MUSCLE`, `CLUSTALW` or `T-Coffee`, especially on nucleotide sequence alignment. This model was also trained to deal with longer sequences, generating MSAs of 5 sequences between 500 and 1000 residues long. In this setting `BetaAlign` performs on par with most widely used aligners.

While `BetaAlign` is an interesting step in the direction of learned alignment methods, and a good proof of concept, it seems to be efficient only on a low number of short sequences. This is mostly due to the attention mechanism at the heart of transformers.

7.2.3. The attention limitation

While the transformer architecture has revolutionized the field of machine translation, and proved to be useful in sequence-related bioinformatics tasks, the attention mechanism at its heart presents some problems. The main problem is that by including a weighted sum of all input tokens in the embedding of a specific token, the time and space complexity of the attention mechanism is quadratic in sequence length. This is particularly problematic in biological tasks where DNA and protein sequences can be much longer than a typical sentence, in any spoken language. This limitation is mentioned in the articles for both the `DEDAL` and `BetaAlign` models described above.

This problem is not inherent to biology and many different approaches to counter it have been proposed in other fields where transformer usage is prevalent. The *Linformer*[728] and *Nyströmformer*[729] architectures both present different approximations of the attention mechanism that scale linearly w.r.t. sequence length both in time and memory. Others yet have tried to make the attention process produce sparse matrices, reducing the memory requirements.[730,731] Others have tried adjusting the attention span, i.e. the number of tokens taken into account in the attention mechanism, with an adaptive attention span[732] or long-short range attention.[733] Finally, with some change the operations in the attention mechanism, the *Reformer* model reduces the memory requirements to a linear complexity by replacing a dot product operation.[734]

Some improvements to the attention mechanism have also been tried in a biological context. Choromanski *et al.* propose the *Performer* model that uses a fast attention mechanism,[735] based on orthogonal random features and trained on an protein MLM task. With this approach, the attention mechanism scales linearly with the sequence length rather than quadratically. Another team used factored attention in their model trained on protein structure prediction.[736] They show that

with this mechanism, fewer parameters need to be tuned, lowering the memory and time requirements, while retaining structurally relevant information.

7.2.4. Predicting read-mappings

In the read-mapping setting, the methods described above are of limited use. This is due to some intrinsic characteristics of read-mapping: mainly the size discrepancy between reads and the reference sequence, as well as the length of the reference sequence. Some work has been done however on including machine learning methods into the read-mapping process.

One first approach is to learn data structures, called *learned indices*, used to store potential seeds in the seed and extend framework. These learned indices are trained to replicate the output of a particular data structure. This approach was first proposed in 2018,[737] although it was not implemented then. The `BWA-MEME`[738] read-mapper uses a learned index that predicts positions in a suffix array. This approach is also the one used by the `Sapling` algorithm.[739] Learned indices have also been used to predict a position in an FM-index.[740] These learned indices lower the memory cost and execution time costs by eliminating the need to build the whole data structure and only storing a reduced amount of information. Furthermore it is well adapted to read-mapping since it only needs to be trained once on a specific reference sequence that can be used anytime reads need to be mapped to this reference.

Another approach where machine learning has proven useful is in learning a seed selection scheme. `DeepMinimizer`[741] is one such method, where neural networks are trained to select appropriate minimizers from a DNA sequence. This approach results in minimizers (c.f. Section 2.2.2.1.2) with optimal density, that is to say they are spread out evenly over the whole sequence lowering the memory and time costs of building a seed index. Similarly, although not a direct application of read mapping, deep learning has been used to predict candidate alignment sites in mRNA-miRNA pairs, a similar task to seed selection.[742]

Finally, the pre-processing function framework of MSRs presented in Chapter 3 could also be extended with machine learning methods. Learning connections in the graphs representing MSRs could allow the exploration of the large function spaces of higher-order MSRs. Alternatively some sequence-to-sequence models like transformers could also be used to learn a pre-processing function. To learn an appropriate pre-processing function in an end-to-end fashion, a differentiable read-mapping algorithm is needed. Differentiable versions of the NW and SW could be used in read-mappers, but differentiable seeding and seed-selection processes are also needed.

7.3. Conclusion

Deep learning is a powerful framework for sequence-based tasks. The recent transformer architecture has shown an unprecedented ability to capture within-sequence dependencies and learn relevant information. This ability has made them dominant in the NLP field, particularly machine translation. Transformers and large language models have shown some power in biological sequence processing and sequence alignment. However, the attention mechanism that makes these models so successful has limitations, especially w.r.t. input sequence length. Some approaches and approximations, have been proposed to lower the time and memory complexity of the attention mechanism, but these improvements have yet to be implemented in a sequence alignment task. In the special case of read-mapping, even with improved attention mechanisms, the size discrepancy between reference and reads, as well as the often very large scale of the reference sequence, make transformer based embedding approaches impractical. Learned data structures and seeding schemes might be one of the approaches to improve read alignment.

Global Conclusion

During my PhD I focused on two separate problems both pertaining to biological sequence data. I focused, on the one hand, on improving long-read mapping performance, and on the the other hand, on searching for drug resistance mutations in a large, annotated, HIV multiple sequence alignment.

Improving read-mapping with MSRs

Homopolymer linked errors are the most common error mode in long-reads for both ONT and PacBio sequencing technologies. A common way to mitigate the deleterious effects these errors have on downstream analyses is by using a pre-processing method on the reads and reference sequence: HPC. We developed a new pre-processing framework, defining transformation functions called streaming reduction functions (SSRs). We show that a subset of 58 of these SSRs, that we call mapping-friendly sequence reductions (MSRs), improve mapping-accuracy on simulated Nanopore long-reads over a whole human genome assembly when compared to HPC or no pre-processing. This improvement in mapping-accuracy is also seen when using whole *D. melanogaster* or *E. coli* genomes as references, and does not come at the cost of fewer mapped reads. We also show that these MSRs improve mapping accuracy over repeated regions of the whole human genome. In very low complexity regions of the genome however, such as centromeres, with short, conserved and widely repeated motifs, any pre-processing function (MSR or HPC) is harmful, and keeping the untransformed sequence data is better for the read-mapping task.

In this work, in order to be able to explore the whole SSR function space, we limited ourselves to what we called order-2 SSRs, which consider all pairs of nucleotides as inputs during the sequence pre-processing procedure. It could be interesting to explore higher order SSRs that consider l-mers of nucleotides as inputs. This however leads to a much larger function space. To be able to explore it efficiently we need more biologically informed ways to restrict it, or a way to formulate this exploration as an optimization problem. This optimization approach might be very useful, but one of the main obstacles is the design of a suitable objective function on the read-mapping problem which should, ideally, be differentiable. Differentiable alignment algorithms exist, however read-mapping methods often use heuristics that can be a challenge to include in a loss function. The optimization approach could also be applied to learn MSRs, either by learning connections in the graph representation of

MSRs, or by learning a pre-processing function using sequence to sequence models like transformers. This approach , while exciting, would also require a carefully designed objective function with differentiability properties.

It would also be interesting to apply these MSRs and see if they generalize to other long-read related tasks like clustering or assembly. To evaluate the impact MSRs have on these tasks, some metrics to assess the quality of the produced outputs are needed. Finally, evaluating these MSRs on real data is needed to get a real-world idea of their applicability and usefulness, however evaluating the improvements MSRs might bring to read-mappings without knowing the ground-truth is a challenge.

Searching for resistance mutations in HIV

The global HIV pandemic has been a major public health issue for the last 40 years, claiming more than 30 million victims. Over the years, many anti-retroviral drugs have been developed, targeting most major proteins that are part of the virus' replication cycle. These drugs have helped make the illness manageable in many situations. However, due to HIV's very high mutation rate, most available drugs quickly induce corresponding resistance mutations in the viral population. This is especially true in lower income countries where the diversity of available treatments is lower than in high-income regions, leading to the emergence of multi-resistant virus strains. This in turn can have severe repercussions on public health where resistant strains can be transmitted and spread through the treatment-naive population. We used several machine learning methods in order to explore the resistance landscape of HIV in the UK and Africa, with the goal to find novel drug resistance mutations. By using a large UK dataset of partial HIV-1 Reverse Transcriptase (RT) sequences, we trained three machine learning algorithms to discriminate treatment-naive from treatment-experienced sequences. The classifiers we used, namely naive Bayes, LASSO-regularized logistic regression and random forest, all have built-in measure that allow us to examine which variables in the input are important to classification. By encoding single mutations as single variables we were able to determine which mutations are used by the classifier models to determine if a sequence was exposed to treatment or not.

In order to find novel resistance mutations we removed all mutations that are known to be associated to drug resistance from the training data. In this setting classifiers were statistically significantly better than random, indicating that the models were picking up on residual resistance-associated signal in the training data. Conversely, when removing sequences that were known to contain resistance mutations from the training data, in addition to known resistance-associated features, the classifiers were no better than random. This indicates that all the residual resistance-associated signal that we previously found is contained in sequences that already have known drug resistance mutations. This would indicate that the mutations we identify from our

trained classifiers are accessory in nature and occur only in conjunction with known drug resistance mutations, and that all the primary mutations directly conferring resistance have most likely been found, which is reassuring from a public health perspective.

We identified 6 novel resistance-associated mutations of RT: L228R, L228H, E203K, I135L, H208Y and D218E. We examined the spatial position of these mutations on a structural model of HIV-1 RT, and observed that they were either close to the active site or the allosteric regulation site targeted by RT inhibitors. Furthermore, we used a simple classifier built from mutations found to be significantly associated with treatment using Fisher tests and correcting for multiple-testing. This simple procedure yielded results with an accuracy on par with the more complex models we also trained. We interpreted this fact to mean that complex epistatic phenomena, where a group of mutations have a bigger effect on resistance than the sum of individual effects, are not at play here.

In order to be sure that the mutations we identified do have some role in drug resistance, they should be experimentally studied. This experimental confirmation can be conducted *in vivo* or *in vitro* to study their mechanism and action w.r.t. to associated drugs. In order to try and confirm these results, or identify more mutations it would also be interesting to conduct this machine learning procedure with a larger dataset and more sensitive methods like deep learning, although some care should be taken when extracting important features from these complex models. Replicating this procedure with more metadata, like viral load, or in restricted groups, like cohorts of patients that have received a specific treatment, could also bring insight in how some of the mutations are related to treatment. Finding sufficiently large datasets filling these conditions might be a challenge. Finally, this approach could also be applied to other viral species, like the Hepatitis C virus, where sequence data is abundant and public health benefits evident.

Final words

In conclusion, I hope by this point you will agree with me that sequence and sequence alignment data is fundamental and one of the most useful data types in bioinformatics analyses. As such, any method to improve the sequence quality, the alignment process or the interpretation of alignments is important. Improving these aspects can help researchers down the line gain more insight, and a more accurate representation, of crucial biological processes. This is especially important with the advent of the "age of pandemics" and tracking by sequencing where very large quantities of sequence data will have to be analyzed quickly and with accuracy, all with high stakes. I hope that, with this work, I have contributed, at least a little, to this field and to the advancement of knowledge.

A. Supporting Information for "Mapping-Friendly Sequence Reductions: Going Beyond Homopolymer Compression"

A.1. "TandemTools" dataset generation

This dataset was obtained by taking a human X chromosome HOR sequence, concatenating it 500 times with added mutations in order to obtain an approximately 1 Mbp long sequence. Then 1200 reads were simulated from the sequence using `nanosim`[318] and assembled using a centromere-tailored pipeline.[743] A 10kbp deletion was then added to this assembly. The resulting sequence is the one we refer to as the "Centromeric sequence".

A.2. MSR performance comparison

Table A.1.: **Comparing performance of MSRs on the whole human genome, whole *Drosophila melanogaster* genome, repeated regions of the whole human genome and synthetic centromeric sequence.** Results using `minimap2`[119] and `winnowmap2`.[120] The number of simulated reads for each reference sequence is given in parentheses and called n. Results are reported for mapq thresholds of 60, 50 and 0. The best performance for each category is highlighted in bold. The percentage difference are computed w.r.t HPC at each given threshold.

	mapq=60		mapq≥ 50		any mapq	
mapping friendly sequence reduction	fraction	error	fraction	error	fraction	error
Whole Drosophila melanogaster genome - `minimap2` (n = 25 764)						
HPC	0.957 +0%	2.27e-03 + 0%	**0.963 +0%**	2.34e-03 + 0%	**0.998 +0%**	1.48e-02 + 0%
raw	**0.958 +0%**	2.27e-03 – 0%	0.962 –0%	2.34e-03 + 0%	0.997 –0%	**1.17e-02 –21%**
MSR_F	0.952 –1%	1.18e-03 – 48%	0.960 –0%	1.37e-03 – 41%	**0.998 +0%**	1.36e-02 – 8%
MSR_E	0.946 –1%	**0 –100%**	0.954 –1%	**0 –100%**	**0.998 +0%**	1.53e-02 + 3%
MSR_P	0.950 –1%	4.90e-04 – 78%	0.957 –1%	8.11e-04 – 65%	**0.998 –0%**	1.39e-02 – 6%
Whole Drosophila melanogaster genome - `winnowmap2` (n = 25 764)						
HPC	0.923 +0%	1.51e-03 + 0%	0.930 +0%	1.59e-03 + 0%	0.989 +0%	1.50e-02 + 0%
raw	**0.949 +3%**	1.92e-03 +27%	**0.954 +3%**	1.99e-03 +26%	**0.995 +1%**	**1.33e-02 –12%**
MSR_F	0.918 –1%	**1.27e-03 –16%**	0.925 –0%	**1.30e-03 –18%**	0.987 –0%	1.37e-02 – 9%
MSR_P	0.905 –2%	1.33e-03 –12%	0.912 –2%	1.53e-03 –3%	0.983 –1%	1.40e-02 – 7%
MSR_E	0.905 –2%	1.42e-03 – 6%	0.912 –2%	1.49e-03 – 6%	0.983 –1%	1.44e-02 – 4%
Synthetic centromeric sequence - `minimap2` (n = 12 673)						
HPC	0.870 +0%	**1.36e-03 + 0%**	0.964 +0%	**1.56e-03 + 0%**	**1.000 +0%**	9.00e-03 + 0%
raw	**0.936 +8%**	1.86e-03 + 36%	**0.984 +2%**	2.09e-03 + 34%	**1.000 +0%**	**4.50e-03 –50%**
MSR_E	0.885 +2%	3.39e-03 +149%	0.962 –0%	3.53e-03 +127%	**1.000 +0%**	1.20e-02 +33%
MSR_F	0.850 –2%	2.04e-03 + 50%	0.968 +0%	2.12e-03 + 36%	**1.000 +0%**	6.63e-03 –26%
MSR_P	0.898 +3%	1.58e-03 + 16%	0.968 +0%	1.79e-03 + 15%	**1.000 +0%**	9.78e-03 + 9%
Synthetic centromeric sequence - `winnowmap2` (n = 12 673)						
HPC	0.775 + 0%	**1.32e-03 + 0%**	**0.822 +0%**	**1.82e-03 + 0%**	0.997 +0%	8.37e-02 + 0%
raw	**0.850 +10%**	2.04e-03 +54%	0.890 +8%	1.95e-03 + 7%	**0.999 +0%**	**4.60e-02 –45%**
MSR_E	0.795 + 2%	2.28e-03 +73%	0.846 +3%	2.52e-03 +38%	0.997 –0%	6.96e-02 –17%
MSR_F	0.820 + 6%	1.83e-03 +38%	0.867 +6%	2.27e-03 +25%	0.997 –0%	5.97e-02 –29%
MSR_P	0.780 + 1%	1.62e-03 +22%	0.829 +1%	2.09e-03 +15%	0.997 –0%	8.65e-02 + 3%
Whole human genome - `minimap2` (n = 655 594)						
HPC	**0.935 +0%**	1.85e-03 + 0%	**0.942 +0%**	1.85e-03 + 0%	**1.000 +0%**	1.46e-02 + 0%
raw	0.921 –1%	1.86e-03 + 0%	0.927 –2%	1.86e-03 + 1%	0.998 –0%	**1.29e-02 –11%**
MSR_E	0.926 –1%	**6.92e-05 –96%**	0.936 –1%	**1.17e-04 –94%**	0.999 –0%	1.76e-02 +20%
MSR_P	0.929 –1%	2.20e-04 –88%	0.938 –0%	4.15e-04 –78%	0.999 –0%	1.55e-02 + 6%
MSR_F	0.930 –1%	1.09e-03 –41%	0.938 –0%	1.29e-03 –30%	**1.000 –0%**	1.51e-02 + 4%
Whole human genome - `winnowmap2` (n = 655 594)						
HPC	0.894 + 0%	1.43e-03 + 0%	0.902 +0%	1.49e-03 + 0%	0.988 +0%	1.92e-02 + 0%
raw	**0.932 + 4%**	1.75e-03 +23%	**0.937 +4%**	1.79e-03 +20%	**0.994 +1%**	**1.43e-02 –26%**
MSR_F	0.874 – 2%	2.81e-04 –80%	0.886 –2%	3.82e-04 –74%	0.984 –0%	1.94e-02 + 1%
MSR_E	0.795 –11%	**6.33e-05 –96%**	0.820 –9%	**8.93e-05 –94%**	0.971 –2%	2.08e-02 + 9%
MSR_P	0.826 – 8%	8.68e-05 –94%	0.845 –6%	1.14e-04 –92%	0.975 –1%	2.11e-02 +10%
Whole Human genome (repeated regions) - `minimap2` (n = 68 811)						
HPC	**0.619 + 0%**	3.29e-04 + 0%	0.656 + 0%	3.10e-04 + 0%	**0.998 +0%**	7.79e-02 + 0%
raw	0.514 -17%	1.98e-04 -40%	0.539 -18%	2.16e-04 -30%	0.981 -2%	**6.69e-02 -14%**
MSR_F	0.601 - 3%	2.18e-04 -34%	0.640 - 2%	2.27e-04 -27%	**0.998 -0%**	8.15e-02 + 5%
MSR_E	0.618 - 0%	1.41e-04 -57%	**0.658 + 0%**	**1.55e-04 -50%**	0.997 -0%	8.23e-02 + 6%
MSR_P	0.616 - 1%	**1.18e-04 -64%**	0.656 + 0%	1.99e-04 -36%	0.997 -0%	8.31e-02 + 7%
Whole Human genome (repeated regions) - `winnowmap2` (n = 68 811)						
HPC	0.525 + 0%	1.24e-03 + 0%	0.557 + 0%	1.49e-03 + 0%	0.950 +0%	1.19e-01 + 0%
raw	**0.648 +23%**	1.26e-03 + 1%	**0.672 +21%**	1.49e-03 + 0%	**0.968 +2%**	**8.09e-02 -32%**
MSR_F	0.482 - 8%	**1.63e-03 +31%**	0.516 - 7%	1.83e-03 +23%	0.940 -1%	1.21e-01 + 2%
MSR_E	0.366 -30%	6.35e-04 -49%	0.405 -27%	**9.32e-04 -37%**	0.911 -4%	1.38e-01 +17%
MSR_P	0.415 -21%	9.45e-04 -24%	0.451 -19%	1.16e-03 -22%	0.920 -3%	1.39e-01 +17%

A.3. Analyzing read origin on whole human genome

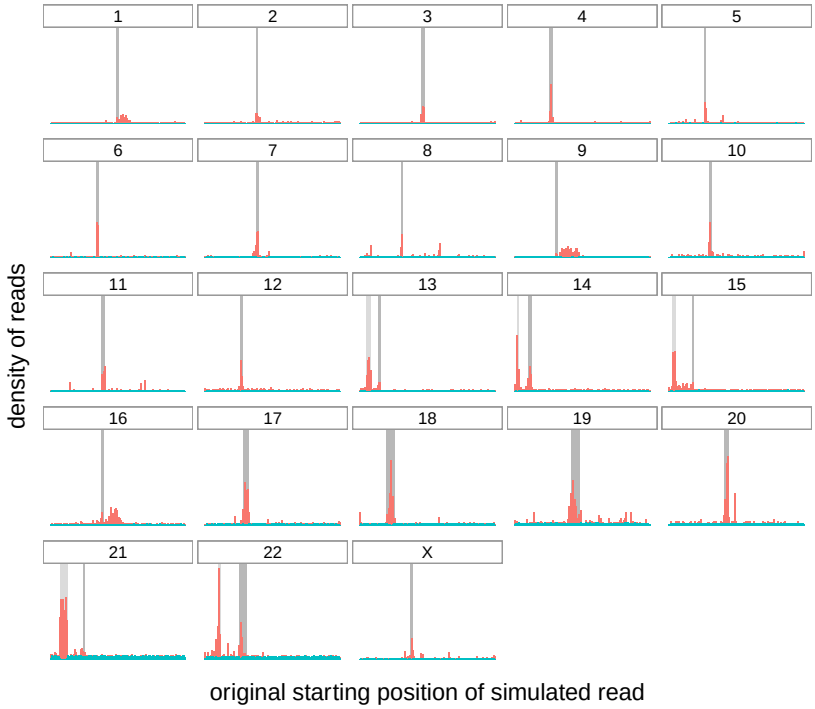


Figure A.1.: **Origin of correctly and incorrectly mapped raw reads.** Distribution of the origin of correctly and incorrectly mapped simulated reads (in teal and red respectively) on the different chromosomes of the whole human genome. The dark grey rectangle for each chromosome represents the centromere of that chromosome. The lighter gray rectangle on chromosomes 13, 14, 15, 21 and 22 correspond to satellites denoted as “stalk”, another repetitive region.

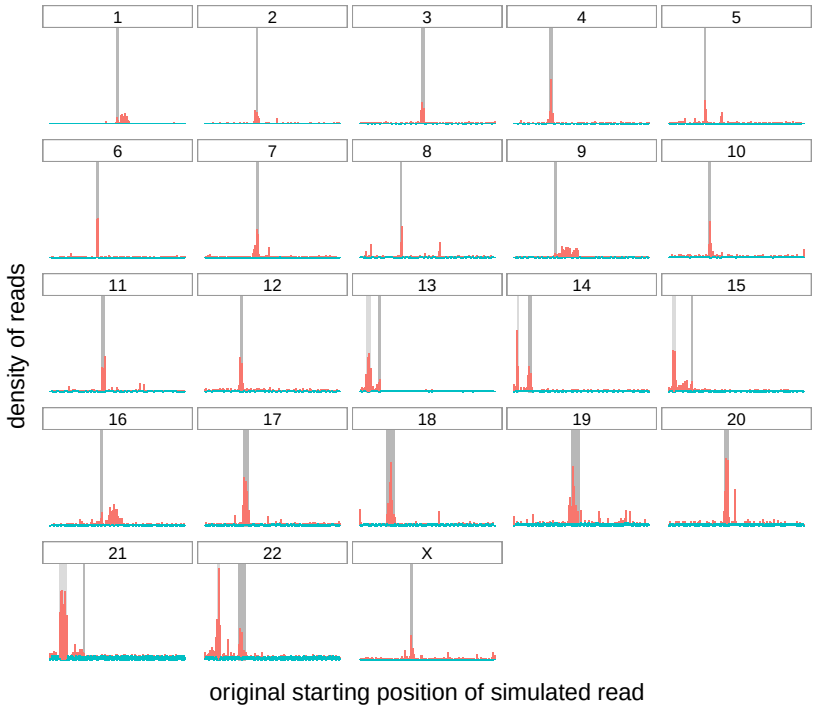


Figure A.2.: **Origin of correctly (teal) and incorrectly (red) mapped reads, transformed with HPC.**
Distribution of the origin of correctly and incorrectly mapped simulated reads (in teal and red respectively) on the different chromosomes of the whole human genome. The dark grey rectangle for each chromosome represents the centromere of that chromosome. The lighter gray rectangle on chromosomes 13, 14, 15, 21 and 22 correspond to satellites denoted as “stalk”, another repetitive region.

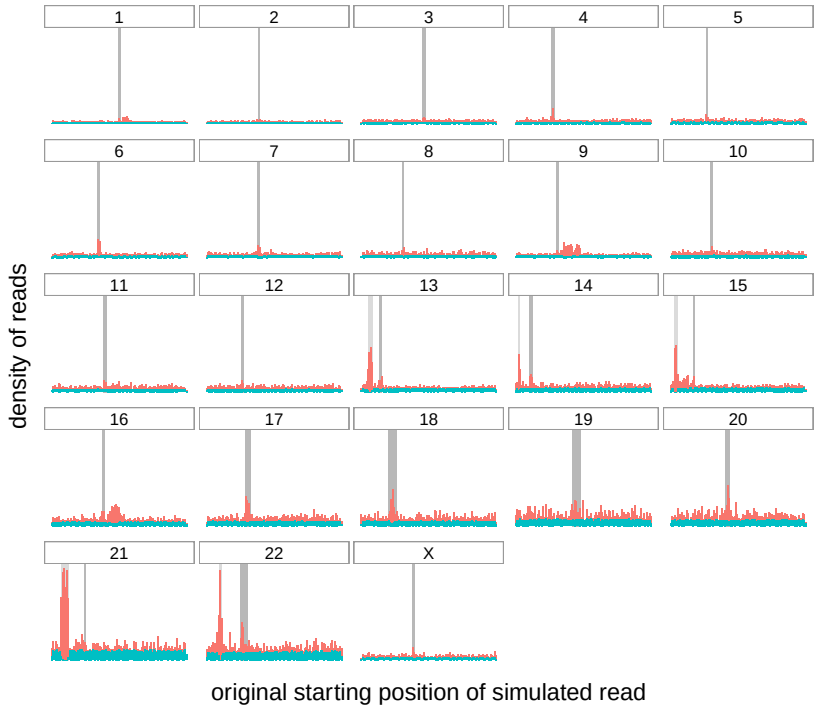


Figure A.3.: **Origin of correctly (teal) and incorrectly (red) mapped reads, transformed with MSR_E.**
Distribution of the origin of correctly and incorrectly mapped simulated reads (in teal and red respectively) on the different chromosomes of the whole human genome. The dark grey rectangle for each chromosome represents the centromere of that chromosome. The lighter gray rectangle on chromosomes 13, 14, 15, 21 and 22 correspond to satellites denoted as “stalk”, another repetitive region.

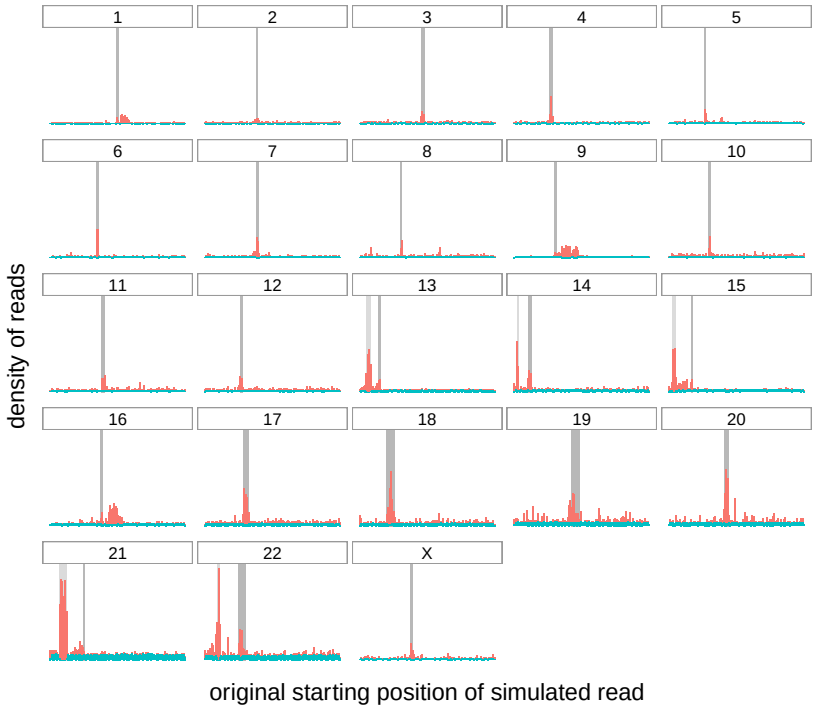


Figure A.4.: **Origin of correctly (teal) and incorrectly (red) mapped reads, transformed with $\mathbf{MSR_P}$.**
Distribution of the origin of correctly and incorrectly mapped simulated reads (in teal and red respectively) on the different chromosomes of the whole human genome. The dark grey rectangle for each chromosome represents the centromere of that chromosome. The lighter gray rectangle on chromosomes 13, 14, 15, 21 and 22 correspond to satellites denoted as “stalk”, another repetitive region.

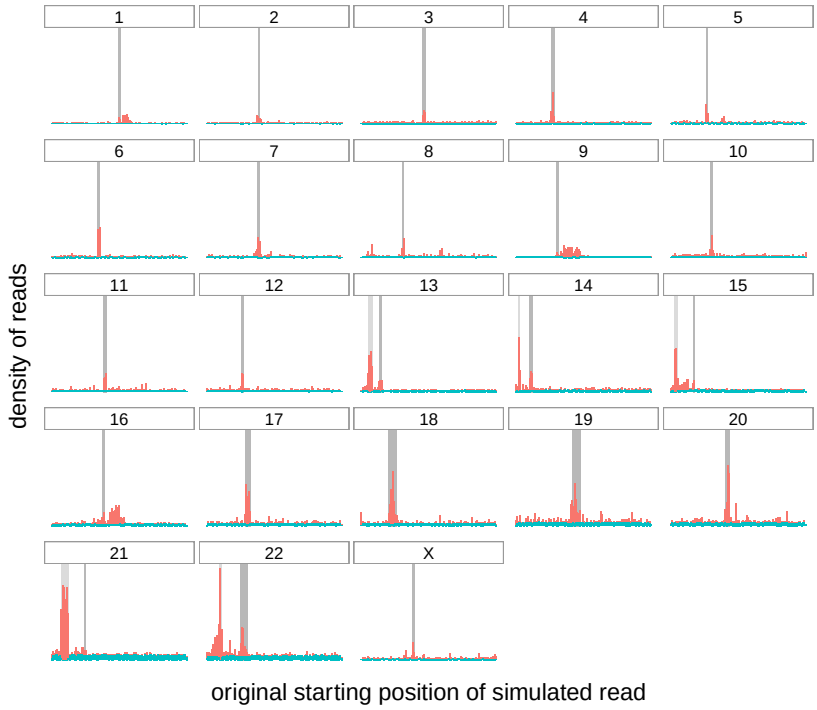


Figure A.5.: **Origin of correctly (teal) and incorrectly (red) mapped reads, transformed with MSR_F.**
Distribution of the origin of correctly and incorrectly mapped simulated reads (in teal and red respectively) on the different chromosomes of the whole human genome. The dark grey rectangle for each chromosome represents the centromere of that chromosome. The lighter gray rectangle on chromosomes 13, 14, 15, 21 and 22 correspond to satellites denoted as “stalk”, another repetitive region.

A.4. Performance of MSRs on the Drosophila genome

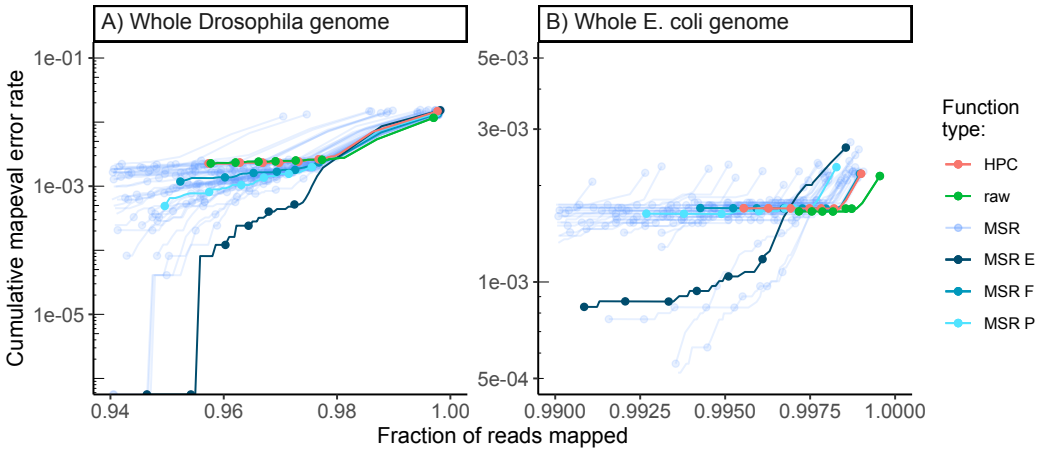


Figure A.6.: **Results of the `paftools mapeval` evaluation on reads simulated and mapped to whole *Drosophila melanogaster* and *Escherichia coli* (Genbank ID U00096.2) genomes.**
MSRs E, F and P are shown in different shades of blue to differentiate them from other MSRs. Reads were simulated with `nanosim`, and mapped with `minimap2`.

A.5. Key resource table

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited Data		
T2T CHM13 v1.1, whole human genome assembly	(Nurk et al., 2022)	Genbank accession number GCA_009914755.3
Release 6 plus ISO1 MT, whole drosophila melanogaster genome assembly	(Adams et al., 2000)	Genbank accession number GCA_000001215.4
Synthetic centrormeric sequence	(Mikheenko et al., 2020)	https://github.com/ablab/TandemTools/blob/master/test_data/simulated_del.fasta
Escherichia coli str. K-12 substr. MG1655, complete genome	(Blattner et al., 1997)	Genbank accession number U00096.2
Coordinates of repeated regions of the CHM13 whole genome assembly	Telomere to Telomere consortium	https://t2t.gi.ucsc.edu/chm13/hub/t2t-chm13-v1.1/rmsk/rmsk.bigBed
Software and Algorithms		
minimap2 v2.22-r1101	(Li, 2018)	https://github.com/lh3/minimap2
Winnowmap v2.0	(Jain et al., 2020)	https://github.com/marbl/Winnowmap
NanoSim v3.0.0	(Yang et al., 2017)	https://github.com/bcgsc/NanoSim
Bedtools v2.30.0	(Quinlan et al., 2010)	https://github.com/arq5x/bedtools2
Meryl v1.0	(Rhie et al., 2020)	https://github.com/marbl/Winnowmap
Analysis pipelines	This paper	https://doi.org/10.5281/zenodo.6859636

B. Supporting Information for "HIV and DRMs"

B.1. Detailed list of HIV-1 protein structures used for figure generation.

The images for HIV-1 structures used in Figure 5.2 were obtained from: https://cdn.rcsb.org/pdb101/learn/resources/structural-biology-of-hiv/. They are licensed under a Creative Commons By 4.0 license which allows reuse and adaptation for non commercial use.

PDB structure IDs:

- **SU** and **TM**: 4nco
- **MA**: 1hiw
- **CA**: 3h47
- **NC**: 1a1t
- **RT**: 1hys (for Figure 5.2) and 2hmi (for Figure 5.4)
- **IN**: 1ex4
- **PR**: 1hpv
- **Vpu**: 1pi7 and 1vpu
- **Vif**: 3dcg
- **Vpr**: 1esx
- **Nef**: 1avv and 1qa5
- **Rev**: 1etf
- **Tat**: 1biv and 1jfw

B.2. List of all antiretroviral drugs

Name	Brand name	Abbreviation	Class	Approval date
zidovudine	retrovir	ZDV	NRTI	1987-03-19
didanosine†	videx	ddI	NRTI	1991-10-09
zalcitabine†	hivid	ddC	NRTI	1992-06-19
stavudine†	zerit	d4T	NRTI	1994-06-24
lamivudine	epivir	3TC	NRTI	1995-11-17
saquinavir	invirase	SQV	PI	1995-12-06
ritonavir*	norvir	RTV	PI	1996-03-01
indinavir†	crixivan	IDV	PI	1996-03-13
neviparine	viramune	NVP	NNRTI	1996-06-21
nelfinavir†	viracept	NFV	PI	1997-03-14
delavirdine†	rescriptor	DLV	NNRTI	1997-04-04
combivir	combivir	3TC+ZDV	FDC	1997-09-27
efavirenz	sustiva	EFV	NNRTI	1998-09-17
abacavir	ziagen	ABC	NRTI	1998-12-17
amprenavir†	agenerase	APV	PI	1999-04-15
kaletra	kaletra	LPV+RTV	FDC	2000-09-15
didanosine-ec†	videx-ec	ddI-EC	NRTI	2000-10-31
trizivir	trizivir	ABC+3TC+ZDV	FDC	2000-11-14
tenofovir-df	viread	TDF	NRTI	2001-10-26
enfuvirtide	fuzeon	T-20	FI	2003-03-13
atazanavir	reyataz	ATC	PI	2003-06-20
emtricitabine	emtriva	FTC	NRTI	2003-07-02
fosamprenavir	lexiva	FPV	PI	2003-10-20
epzicom	epzicom	ABC+3TC	FDC	2004-08-02
truvada	truvada	FTC+TDF	FDC	2004-08-02
tipranavir	aptivus	TPV	PI	2005-06-22
darunavir	prezista	DRV	PI	2006-06-23
atripla	atripla	EFV+FTC+TDF	FDC	2006-07-12
maraviroc	selzentry	MVC	CA	2007-08-06
raltegravir	isentress	RAL	INSTI	2007-10-12
etravirine	intelence	ETR	NNRTI	2008-01-18
neviparine-xr	viramune-xr	NVP-XR	NNRTI	2011-03-25
rilpivirine	edurant	RPV	NNRTI	2011-05-20
complera	complera	FTC+RPV+TDF	FDC	2011-08-10
stribild	stribild	EVG+COBI+FTC+TDF	FDC	2012-08-27
dolutegravir	tivicay	DTG	INSTI	2013-08-12
triumeq	triumeq	ABC+DTG+3TC	FDC	2014-08-22
elvitegravir†	vitekta	EVG	INSTI	2014-09-14
cobicistat	tybost	COBI	PE	2014-09-24
evotaz	evotaz	ATV+COBI	FDC	2015-01-29
prezcobix	prezcobix	DRV+COBI	FDC	2015-01-29

genvoya	genvoya	EVG+COBI+FTC+TAF	FDC	2015-11-05
odefsey	odefsey	FTC+RPV+TAF	FDC	2016-03-01
descovy	descovy	FTC+TAF	FDC	2016-04-04
raltegravir	isentress-hd	RAL	INSTI	2017-05-26
juluca	juluca	DTG+RPV	FDC	2017-11-21
symfi-lo	symfi-lo	EFV+3TC+TDF	FDC	2018-02-05
biktarvy	biktarvy	BIC+FTC+TAF	FDC	2018-02-07
cimduo	cimduo	3TC+TDF	FDC	2018-02-28
ibalizumab-uiyk	trogarzo	TNX-355	PAI	2018-03-06
symfi	symfi	EFV+3TC+TDF	FDC	2018-03-22
symtuza	symtuza	DRV+COBI+FTC+TAF	FDC	2018-07-17
delstrigo	delstrigo	DOR+3TC+TDF	FDC	2018-08-30
doravirine	pifeltro	DOR	NNRTI	2018-08-30
temixys	temixys	3TC+TDF	FDC	2018-11-16
dovato	dovato	DTG+3TC	FDC	2019-04-08
dolutegravir	tivicay-pd	DTG	INSTI	2020-06-12
fostemsavir	rukobia	FTR	AI	2020-07-02
cabenuva	cabenuva	CAB+RPV	FDC	2021-01-22
cabotegravir	vocabria	CAB	INSTI	2021-01-22

Table B.1.: **List of all antiretroviral drugs used in HIV therapy.**
Zidovudine (ZDV) is also referred to as Azidothymidine (AZT) in litterature, Fixed Dose combinations (i.e. single pills combining multiple drugs) are referred to by their commercial name, the composition of these can be seen in the abbreviation. Drugs were ordered by FDA approval date.
AI: Attachment Inhibitor, **CA**: CCR5 Antagonist, **FDC**: Fixed Dose Combination, **FI**: Fusion Inhibitor, **INSTI**: Integrase Inhibitor **NNRTI**: Non-Nucleoside Reverse Transcriptase Inhibitor, **NRTI**: Nucleoside Reverse Transcriptase Inhibitor, **PE**: Pharmacokinetic Enhancer, **PAI**: Post-Attachment Inhibitor, **PI**: Protease Inhibitor. **AI**, **CA**, **FI** and **PAI** can be grouped in a class of Entry inhibitors.
* Although Ritonavir is originally a PI it is now mainly used as a PE to boost the action of other drugs.
†These drugs are no longer available or recommended in HIV treatment guidelines. They may still be used in FDC regimens.
Adapted from https://hivinfo.nih.gov/understanding-hiv/infographics/fda-approval-hiv-medicines and https://hivinfo.nih.gov/understanding-hiv/fact-sheets/fda-approved-hiv-medicines

C. Supporting Information for "Using Machine Learning and Big Data to Explore the Drug Resistance Landscape in HIV"

C.1. S1 Appendix (Technical appendix).

C.1.1. Data

C.1.1.1. Data Availability

The policy of the UK HIV Drug Resistance Database is to make DNA sequences available to any bona fide researcher who submits a scientifically robust proposal, provided data exchange complies with Information Governance and Data Security Policies in all the relevant countries. This includes replication of findings from published studies, although the researcher would be encouraged to work with the main author of the published paper to understand the nuances of the data. Enquiries should be addressed to iph.hivrdb@ucl.ac.uk in the first instance. More information on the UK dataset is also available on the UK CHIC homepage: www.ukchic.org.uk. Amino acid sequences are made available along with a metadata file.
The West and central African dataset is available as supplementary information along with a metadata file containing HIV subtype, treatment information and known RAM presence/absence for each sequence.
Predictions made for each sequence of both datasets, by all of the trained classifiers are made available as part of the supplementary data as well as synthetic results from which the figures of the paper were drawn. The importance values for each mutation and each trained classifier are also made available.
All the data and metadata files made available are hosted in the online repository linked to this project at the following URL:
github.com/lucblassel/HIV-DRM-machine-learning/tree/main/data

C.1.1.2. Data Preprocessing

For both the African and UK datasets, the sequences were truncated to keep sites 41 to 235 of the RT protein sequence before encoding. This truncation was needed to avoid the perturbation to classifier training due to long gappy regions at the beginning and end of the UK RT alignment caused by shorter sequences. These positions were determined with the Gblocks software[744] with default parameters, except for the Maximum number of sequences for a flanking position, set to 50,000, and the Allowed gap positions, which was set to "All". The encoding was done with the `OneHotEncoder` from the category-encoders python module.[409]

C.1.2. Classifiers

We used classifier implementations from the scikit-learn python library,[745] `RandomForestClassifier` for the random forest classifier, `MultinomialNB` for Naïve Bayes and `LogisticRegressionCV` for logistic regression.
`RandomForestClassifier` was used with default parameters except:

- `"n_jobs"=4`
- `"n_estimators"=5000`

`LogisticRegressionCV` was used with the following parameters:

- `"n_jobs"=4`
- `"cv"=10`
- `"Cs"=100`
- `"penalty"='l1'`
- `"multi_class"='multinomial'`
- `"solver"='saga'`
- `"scoring"='balanced_accuracy'`

`MultinomialNB` was used with default parameters.

For the Fisher exact tests, we used the implementation from the scipy python library,[746] and corrected p-values for multiple testing with the statsmodels python library[747] using the `"Bonferroni"` method.

C.1.3. Scoring

To evaluate classifier performance several measures were used. We computed balanced accuracy instead of classical accuracy, because it can be overly optimistic, especially when assessing a highly biased classifier on an unbalanced test set.[380]The balanced accuracy is computed using the following formula, where TP and TN are the number of true positives and true negatives respectively, and FP and FN are

the number of false positives and false negatives respectively:

$$balanced\ accuracy = \frac{1}{2}\left(\frac{TP}{TP+FP} + \frac{TN}{TN+FN}\right)$$

We also computed adjusted mutual information (AMI). We chose it over mutual information (MI) because it has an upper bound of 1 for a perfect classifier and is not dependent on the size of the test set, allowing us to compare the performance for differently sized test sets.[150] The adjusted mutual information of variables U and V is defined by the following formula, where $MI(U,V)$ is the mutual information between variables U and V, $H(X)$ is the entropy of the variable X ($= U$ or V) and $E\{MI(U,V)\}$ is the expected MI, as explained in.[748]

$$AMI(U,V) = \frac{MI(U,V) - E\{MI(U,V)\}}{\frac{1}{2}[H(U)+H(V)] - E\{MI(U,V)\}}$$

MI was used to compute the G statistic, which follows the chi-square distribution under the null hypothesis.[749] This was used to compute p-values for each of our classifiers and assess the significance of their performance. G is defined by equation below, where N is the number of samples.

$$G = 2 \cdot N \cdot MI(U,V)$$

Finally, to check the probabilistic predictive power of the classifiers we also computed the Brier score which is the mean squared difference between the ground truth and the predicted probability of being of the positive class for every sequence in the test set (therefore lower is better for this metric). The Brier score is defined in equation below, where p_t is the predicted probability of being of the positive class for sample t and o_t is the actual class (0 or 1, 1=positive class) of sample t:

$$Brier\ score = \frac{1}{N}\sum_{t=1}^{N}(p_t - o_t)^2$$

We used the following implementations from the scikit-learn python library[745] with default options:

- `balanced_accuracy_score`
- `mutual_info_score`
- `adjusted_mutual_info_score`
- `brier_score_loss`

We used the relative risk to observe the relationship between one of our new mutations and a binary character X such as treatment status or presence/absence of a

known RAM.

$$RR(new, X) = \frac{prevalence\,(new\ mutation \mid X = 1)}{prevalence\,(new\ mutation \mid X = 0)}$$

$$= \frac{|(new = 1) \cap (X = 1)|}{|(X = 1)|} \div \frac{|(new = 1) \cap (X = 0)|}{|(X = 0)|}$$

C.2. S1 Fig.

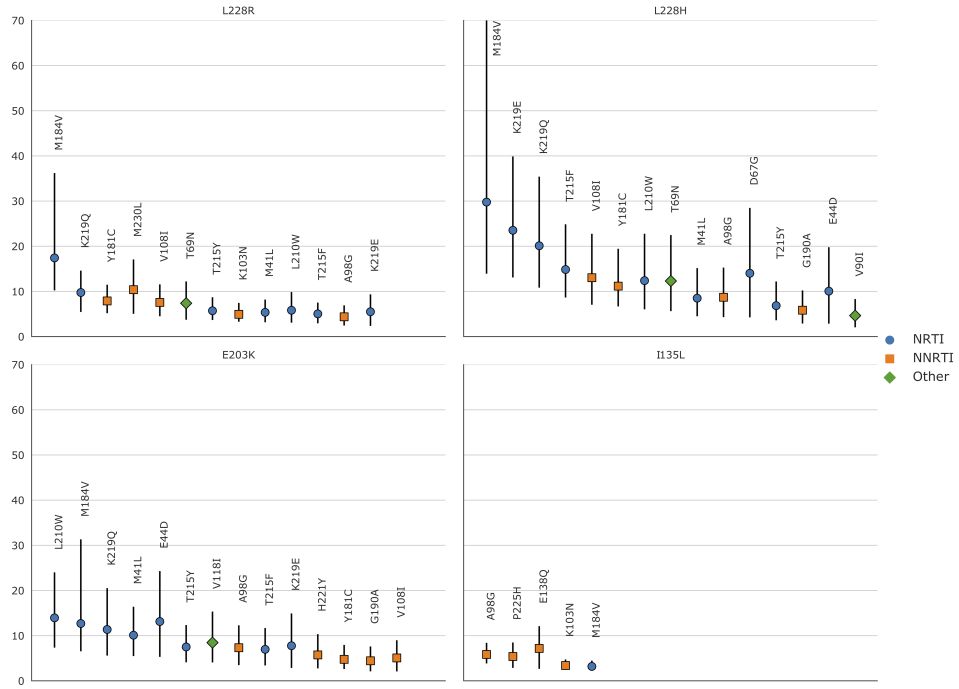


Figure C.1.: **Relative risks of the new mutations with regards to known RAMs on the African dataset.**
(i.e. the prevalence of the new mutation in sequences with a given RAM divided by the prevalence of the new mutation in sequences without the RAM). RRs were only computed for mutations (new and RAMs) that appeared in at least 30 sequences, which is why RRs were not computed for H208Y and D218E. 95% confidence intervals, represented by vertical bars, were computed with 1000 bootstrap samples of the African sequences. Only RRs with a lower CI boundary greater than 2 are shown. The shape and color of the point represents the type of RAM as defined by Stanford's HIVDB. Blue circle: NRTI, orange square: NNRTI, green diamond: Other. For the RR of L228H with regards to M184V, the upper CI bound is infinite. The new RAMs have high RR values for known RAMs similar to those obtained on the UK dataset. We also arrive at similar conclusions, I135L being associated with NNRTIs, E203K and L228H to NRTI and L228R to both. RR values are shown from left to right, by order of decreasing values on the lower bound of the 95% CI.

C.3. S2 Fig.

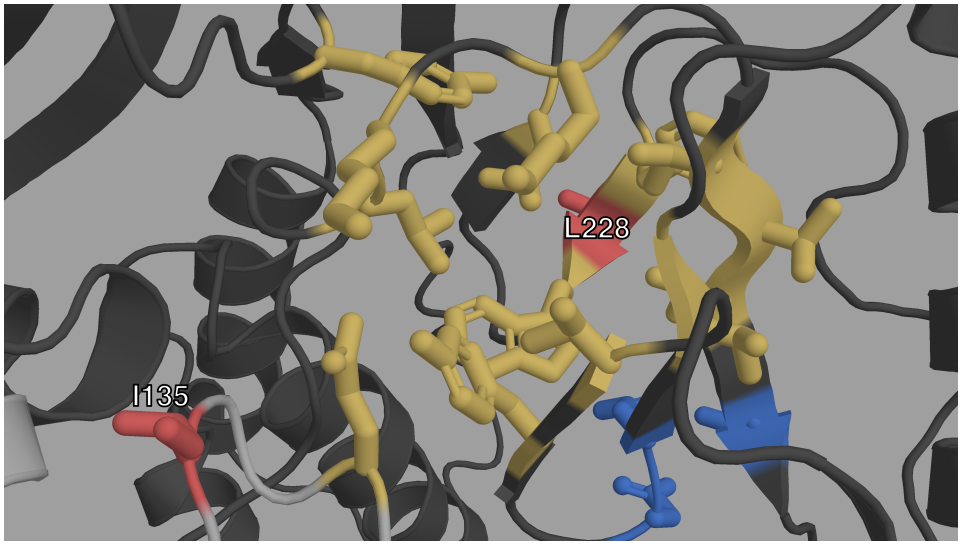


Figure C.2.: **Closeup structural view of the entrance of the NNIBP of HIV-1 RT.**
The p66 subunit is colored in dark gray, the p51 subunit in light gray. The NNIBP is highlighted in yellow. The active site is colored in blue. We can see the physical proximity of I135 (red) to the entrance of the NNIBP. We can also see how L228 (red) is between 2 AAs of the NNIBP.

C.4. S3 Fig.

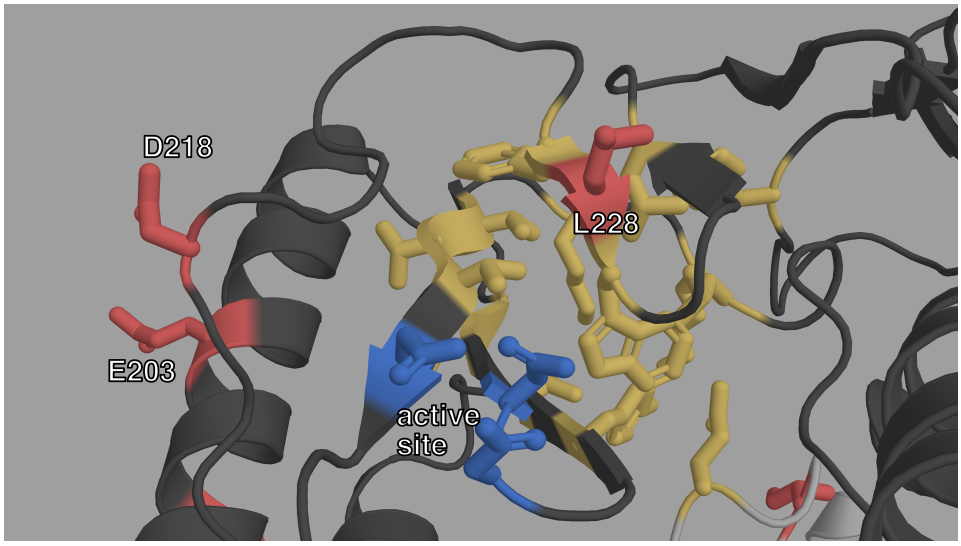


Figure C.3.: **Closeup structural view of the active site of HIV-1 RT.** The p66 subunit is colored in dark gray, the p51 subunit in light gray. The active site is highlighted in blue. The NNIBP is colored in yellow. L228, E203 and D218 (red) are also very close on either side of the active site.

C.5. S1 Table.

	rank		codon distance			UK			Africa									
	T/N	W/W	min	UK	Africa	count	ratio $\rho(new, treatment)$	$\rho(new, with\ RAM)$	count	ratio $\rho(new, treatment)$	$\rho(new, with\ RAM)$	p-value	B62	Dayhoff category shift	Change in net charge	polarity	hydrophobicity index	molecular weight
L228R	0	0	1	1.16	1.21	227 (0.4%)	18.1 [12.9;27.3]	115.7 [55.1;507.3]	98 (2.5%)	32.5[15.4;147.1]	42.4 [17.8;∞]	2.0E-30	-2	e → d	1	5.6	-0.93	43.03
E203K	1	1	1	1.31	1.33	256 (0.5%)	11.0 [8.2;15.1]	20.1 [13.7;32.1]	56 (1.4%)	14.1[6.7;71.9]	17.4 [8.2;83.7]	6.4E-14	1	c → d	2	-1	0.68	-0.94
D218E	2	3	1	1	1	168 (0.3%)	13.1 [9.0;19.6]	27.0 [16.3;57.0]	25 (0.6%)	∞ [∞;∞]	∞ [∞;∞]	2.0E-09	2	c → c	0	-0.7	0.01	14.03
L228H	3	4	1	1.12	1.17	287 (0.5%)	6.4 [5.1;8.4]	9.2 [6.9;12.6]	53 (1.3%)	23.1[9.4;∞]	34.1 [12.0;∞]	2.7E-15	-3	e → d	0	5.5	-0.92	23.99
I135L	4	6	1	1.16	1.13	540 (1.0%)	1.8 [1.5;2.1]	2.4 [2.0;2.8]	134(3.4%)	2.6 [1.8;3.8]	2.4 [1.7;3.4]	2.6E-07	2	e → e	0	-0.3	-0.69	0
H208Y	8	9	1	1.10	1.12	205 (0.4%)	8.8 [6.5;12.5]	14.9 [9.9;23.6]	13 (0.3%)	∞ [∞;∞]	∞ [∞;∞]	7.3E-05	2	d → f	0	-4.2	1.27	26.03

Table C.1.: **Detailed view of the characteristics of new potential RAMs.**
Rank: For each new mutation we computed the aggregate feature importance ranks for the RTI-naive / RTI-experienced and known RAM present / known RAM absent classification tasks. **Codon distance:** We computed the minimum number of nucleotide mutations to go from the wild amino acid codons to those of the mutated amino acid, as well as the average codon distance between both amino acids, weighted by the prevalence of each wild and mutated codon in the UK and the African datasets. **Count (both UK and Africa):** We looked at the number of apparitions of each new potential RAM in the UK and African datasets and the corresponding prevalence in parentheses. **Ratio (both UK and Africa):** We computed the prevalence ratio $\rho(new, treatment)$ (e.g. L228R is 18.1 times more prevalent in RTI-experienced sequences compared to RTI-naive sequences in the UK dataset). We also computed the prevalence ratio $\rho(new, anyRAM)$ (e.g. L228R is 115.7 times more prevalent in sequences that have at least one known RAM than in sequences that have none in the UK dataset). The 95% confidence intervals shown under each ratio were computed with 1000 bootstrap samples of size $n = 55,000$ drawn with replacement from the whole UK dataset (The same procedure was done on the African dataset with size $n = 3990$). **p-values:** Fisher exact tests were done on the African dataset to see if each of these new mutations were more prevalent in RTI-experienced sequences; p-value were corrected with the Bonferroni method for the six simultaneous tests. **B62:** BLOSUM62 similarity values (e.g. D218E = 2, reflecting that E and D are both negatively charged and highly similar). **Dayhoff category shift:** The change in Dayhoff amino acid category is written thusly: "starting category → ending category". These categories are as follows: *a:* Sulfur polymerization. *b:* Small, *c:* Acid and amide, *d:* Basic, *e:* Hydrophobic and *f:* aromatic. **Physico-chemical change:** Change in physicochemical properties was obtained by subtracting the property value of the wil-type amino acid from the mutated amino acid. All values were obtained from the AAindex database[410]

C.6. S2 Appendix. (Fisher exact tests)

Fisher exact tests on pairs of mutations. A detailed explanation of the procedure followed to test pairs of mutations for association with treatment. Detailed numerical results are also given.

In order to study epistasis further we conducted conducted Fisher exact tests between every pair of mutations in the UK dataset ($n = 867,903$) and the treatment status, corrected the p-values with the Bonferroni method with an overall risk level $\alpha = 0.05$.
Out of these tests, $1,309$ pairs were significantly associated with treatment status. 424 out of $1,309$ these pairs were two known RAMs, 806 of these pairs contained one known RAM and only 79 tests had pairs involving no known RAM at all. Furthermore out of these $1,309$ significantly associated pairs, 829 contained two mutations that were significantly associated to treatment when testing mutations one by one. In 478 pairs, one of the two mutations is associated to treatment on its own, and the remaining 2 pairs, none of the mutations were significantly associated with treatment on their own.
These 2 pairs were K103R + V179D and T165I + K173Q. The first pair, is a pair of known RAMs and this interaction is characterized in the HIVDb database (https://hivdb.stanford.edu/dr-summary/comments/NNRTI/). The second pair is made up of new mutations, and the corrected p-value is 0.02. In the Standford HIVDB, T165I has been associated to a reduction in EFV susceptibility.
Out of the $1,309$ pairs significantly associated to treatment, 151 contained at least one of our 6 new potential RAMs, in 6 cases the pair was made up of 2 of them.
In the UK dataset, phylogenetic correlation is likely very impactful with regards to these tests. Indeed, the sequences are far from being independent. In order to alleviate this effect we decided to test the sigficative pairs again on the African dataset, and once more correct with the Bonferroni procedure.
Out of the $1,309$ tests 294 have significative p-values after correction. Out of these 221 pairs were composed of 2 mutations individually significatively associated with treatment. The remaining 73 pairs had one mutation significantly associated with treatment.
Out of the 221 significative tests, 156 pairs were composed of 2 known RAMS while 135 had one known RAM in the pair. The remaining 3 pairs that do not contain a known RAM all contained either L228R or L228H which are both part of our 6 potential RAMS.

C.7. S1 Data.

Archive of figure generating data. A zip archive containing the processed data used to generate each panel of the main figures.

https://doi.org/10.1371/journal.pcbi.1008873.s007 (ZIP)

C.8. S2 Data.

List of known DRMs. A .csv file containing all the known RAMs used in this project as well as the corresponding feature name in the encoded datasets. Obtained from (hivdb.stanford.edu/dr-summary/comments/NRTI/) and (hivdb.stanford.edu/dr-summary/comments/NNRTI/).

https://doi.org/10.1371/journal.pcbi.1008873.s008 (CSV)

Global References

1. J. D. Watson and F. H. C. Crick: The Structure of Dna. *Cold Spring Harbor Symposia on Quantitative Biology* **18** (Jan. 1, 1953), 123–131. doi: 10.1101/sqb.1953.018.01.020 (cit. on p. 5).
2. F. Sanger et al.: Nucleotide Sequence of Bacteriophage X174 DNA. *Nature* **265**(5596) (5596 Feb. 1977), 687–695. doi: 10.1038/265687a0 (cit. on pp. 5, 11).
3. C. T. Archer et al.: The Genome Sequence of E. Coli W (ATCC 9637): Comparative Genome Analysis and an Improved Genome-Scale Reconstruction of E. Coli. *BMC Genomics* **12**(1) (Jan. 6, 2011), 9. doi: 10.1186/1471-2164-12-9 (cit. on p. 5).
4. S. Nurk et al.: The Complete Sequence of a Human Genome. *Science* **376**(6588) (Apr. 2022), 44–53. doi: 10.1126/science.abj6987 (cit. on pp. 5, 14, 17, 18, 21, 48, 55, 56, 64).
5. J. Pellicer, M. F. Fay, and I. J. Leitch: The Largest Eukaryotic Genome of Them All? *Botanical Journal of the Linnean Society* **164**(1) (Sept. 1, 2010), 10–15. doi: 10.1111/j.1095-8339.2010.01072.x (cit. on p. 5).
6. H. C. Macgregor: C-Value Paradox. *Encyclopedia of Genetics*. New York, Jan. 1, 2001, 249–250. doi: 10.1006/rwgn.2001.0301 (cit. on p. 6).
7. B. Alberts et al.: *Molecular Biology of the Cell. 4th Edition*. 2002. https://www.ncbi.nlm.nih.gov/books/NBK26916/ (cit. on p. 6).
8. F. H. C. Crick, L. Barnett, S. Brenner, and R. J. Watts-Tobin: General Nature of the Genetic Code for Proteins. *Nature* **192**(4809) (4809 Dec. 1961), 1227–1232. doi: 10.1038/1921227a0 (cit. on p. 6).
9. International Human Genome Sequencing Consortium: Finishing the Euchromatic Sequence of the Human Genome. *Nature* **431**(7011) (7011 Oct. 2004), 931–945. doi: 10.1038/nature03001 (cit. on p. 7).
10. R. Elkon and R. Agami: Characterization of Noncoding Regulatory DNA in the Human Genome. *Nature Biotechnology* **35**(8) (8 Aug. 2017), 732–746. doi: 10.1038/nbt.3863 (cit. on p. 7).
11. G. S. Omenn: Reflections on the HUPO Human Proteome Project, the Flagship Project of the Human Proteome Organization, at 10Yearš. *Molecular & Cellular Proteomics : MCP* **20** (Feb. 26, 2021), 100062. doi: 10.1016/j.mcpro.2021.100062 (cit. on p. 7).
12. S. A. Shabalina and N. A. Spiridonov: The Mammalian Transcriptome and the Function of Non-Coding DNA Sequences. *Genome Biology* **5**(4) (Mar. 25, 2004), 105. doi: 10.1186/gb-2004-5-4-105 (cit. on p. 7).
13. T. E. P. Consortium: An Integrated Encyclopedia of DNA Elements in the Human Genome. *Nature* **489**(7414) (Sept. 6, 2012), 57–74. doi: 10.1038/nature11247 (cit. on p. 7).

14. N. Chatterjee and G. C. Walker: Mechanisms of DNA Damage, Repair, and Mutagenesis: DNA Damage and Repair. *Environmental and Molecular Mutagenesis* **58**(5) (June 2017), 235–263. doi: 10.1002/em.22087 (cit. on p. 7).

15. I. J. Fijalkowska, R. M. Schaaper, and P. Jonczyk: DNA Replication Fidelity in Escherichia Coli: A Multi-DNA Polymerase Affair. *FEMS microbiology reviews* **36**(6) (Nov. 2012), 1105–1121. doi: 10.1111/j.1574-6976.2012.00338.x (cit. on p. 7).

16. L. Pray: DNA Replication and Causes of Mutation. *Nature education* **1**(1) (2008), 214. https://www.nature.com/scitable/topicpage/dna-replication-and-causes-of-mutation-409/ (cit. on p. 7).

17. J.-F. Gout, W. K. Thomas, Z. Smith, K. Okamoto, and M. Lynch: Large-Scale Detection of in Vivo Transcription Errors. *Proceedings of the National Academy of Sciences* **110**(46) (Nov. 12, 2013), 18584–18589. doi: 10.1073/pnas.1309843110 (cit. on p. 8).

18. J.-F. Gout et al.: The Landscape of Transcription Errors in Eukaryotic Cells. *Science Advances* **3**(10) (Oct. 20, 2017), e1701484. doi: 10.1126/sciadv.1701484 (cit. on p. 8).

19. D. Shcherbakov et al.: Ribosomal Mistranslation Leads to Silencing of the Unfolded Protein Response and Increased Mitochondrial Biogenesis. *Communications Biology* **2**(1) (1 Oct. 17, 2019), 1–16. doi: 10.1038/s42003-019-0626-9 (cit. on p. 8).

20. O. Desouky, N. Ding, and G. Zhou: Targeted and Non-Targeted Effects of Ionizing Radiation. *Journal of Radiation Research and Applied Sciences* **8**(2) (Apr. 1, 2015), 247–254. doi: 10.1016/j.jrras.2015.03.003 (cit. on p. 8).

21. J. Kiefer: Effects of Ultraviolet Radiation on DNA. *Chromosomal Alterations: Methods, Results and Importance in Human Health*. Berlin, Heidelberg, 2007, 39–53. doi: 10.1007/978-3-540-71414-9_3 (cit. on p. 8).

22. J. W. Bennett and M. Klich: Mycotoxins. *Clinical Microbiology Reviews* **16**(3) (July 2003), 497–516. doi: 10.1128/cmr.16.3.497-516.2003 (cit. on p. 8).

23. O. Kantidze, A. Velichko, A. Luzhin, and S. Razin: Heat Stress-Induced DNA Damage. *Acta Naturae* **8**(2) (2016), 75–78. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4947990/ (cit. on p. 8).

24. C. D. Gregory and A. E. Milner: Regulation of Cell Survival in Burkitt Lymphoma: Implications from Studies of Apoptosis Following Cold-Shock Treatment. *International Journal of Cancer* **57**(3) (May 1, 1994), 419–426. doi: 10.1002/ijc.2910570321 (cit. on p. 8).

25. A. Gafter-Gvili et al.: Oxidative Stress-Induced DNA Damage and Repair in Human Peripheral Blood Mononuclear Cells: Protective Role of Hemoglobin. *PLoS ONE* **8**(7) (July 9, 2013), e68341. doi: 10.1371/journal.pone.0068341 (cit. on p. 8).

26. M. Anagnostou et al.: Transcription errors in aging and disease. *Translational Medicine of Aging* **5** (2021), 31–38. doi: 10.1016/j.tma.2021.05.002 (cit. on p. 8).

27. J. R. Roth: Frameshift Mutations. *Annual Review of Genetics* **8** (1974), 319–346. doi: 10.1146/annurev.ge.08.120174.001535 (cit. on p. 8).

28. J. L. Kujovich: Factor V Leiden Thrombophilia. *Genetics in Medicine* **13**(1) (Jan. 1, 2011), 1–16. doi: 10.1097/GIM.0b013e3181faa0f2 (cit. on p. 9).

29. G. R. Cutting: Cystic Fibrosis Genetics: From Molecular Understanding to Clinical Application. *Nature Reviews Genetics* **16**(1) (1 Jan. 2015), 45–56. doi: 10.1038/nrg3849 (cit. on p. 9).

30. C. Fuchsberger et al.: The Genetic Architecture of Type 2 Diabetes. *Nature* **536**(7614) (7614 Aug. 2016), 41–47. doi: 10.1038/nature18642 (cit. on p. 9).

31. A. P. Morris et al.: Large-Scale Association Analysis Provides Insights into the Genetic Architecture and Pathophysiology of Type 2 Diabetes. *Nature genetics* **44**(9) (Sept. 2012), 981–990. doi: 10.1038/ng.2383 (cit. on p. 9).

32. N. Woodford and M. J. Ellington: The Emergence of Antibiotic Resistance by Mutation. *Clinical Microbiology and Infection* **13**(1) (Jan. 1, 2007), 5–18. doi: 10.1111/j.1469-0691.2006.01492.x (cit. on p. 9).

33. S.-Y. Rhee et al.: Human immunodeficiency virus reverse transcriptase and protease sequence database. *Nucleic Acids Research* **31**(1) (Jan. 1, 2003), 298–303. doi: 10.1093/nar/gkg100 (cit. on pp. 9, 99).

34. F. Sanger, S. Nicklen, and A. R. Coulson: DNA Sequencing with Chain-Terminating Inhibitors. *Proceedings of the National Academy of Sciences* **74**(12) (Dec. 1977), 5463–5467. doi: 10.1073/pnas.74.12.5463 (cit. on p. 10).

35. L. M. Smith, S. Fung, M. W. Hunkapiller, T. J. Hunkapiller, and L. E. Hood: The Synthesis of Oligonucleotides Containing an Aliphatic Amino Group at the 5 Terminus: Synthesis of Fluorescent DNA Primers for Use in DNA Sequence Analysis. *Nucleic Acids Research* **13**(7) (Apr. 11, 1985), 2399–2412. doi: 10.1093/nar/13.7.2399 (cit. on p. 11).

36. L. M. Smith et al.: Fluorescence Detection in Automated DNA Sequence Analysis. *Nature* **321**(6071) (6071 June 1986), 674–679. doi: 10.1038/321674a0 (cit. on p. 11).

37. W. Ansorge, B. Sproat, J. Stegemann, C. Schwager, and M. Zenke: Automated DNA Sequencing: Ultrasensitive Detection of Fluorescent Bands during Electrophoresis. *Nucleic Acids Research* **15**(11) (June 11, 1987), 4593–4602. doi: 10.1093/nar/15.11.4593 (cit. on p. 11).

38. J. Shendure and H. Ji: Next-Generation DNA Sequencing. *Nature Biotechnology* **26**(10) (10 Oct. 2008), 1135–1145. doi: 10.1038/nbt1486 (cit. on pp. 11, 12).

39. F. S. Collins, M. Morgan, and A. Patrinos: The Human Genome Project: Lessons from Large-Scale Biology. *Science* **300**(5617) (Apr. 11, 2003), 286–290. doi: 10.1126/science.1084564 (cit. on p. 12).

40. L. Liu et al.: Comparison of Next-Generation Sequencing Systems. *Journal of Biomedicine and Biotechnology* **2012** (July 5, 2012), e251364. doi: 10.1155/2012/251364 (cit. on pp. 12, 14).

41. *The Cost of Sequencing a Human Genome*. Genome.gov. https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost (cit. on p. 12).

42. M. L. Metzker: Sequencing Technologies — the next Generation. *Nature Reviews Genetics* **11**(1) (1 Jan. 2010), 31–46. doi: 10.1038/nrg2626 (cit. on p. 12).

43. B. Canard and R. S. Sarfati: DNA Polymerase Fluorescent Substrates with Reversible 3 -Tags. *Gene* **148**(1) (Oct. 11, 1994), 1–6. doi: 10.1016/0378-1119(94)90226-7 (cit. on p. 13).

44. P. Nyren, B. Pettersson, and M. Uhlen: Solid Phase DNA Minisequencing by an Enzymatic Luminometric Inorganic Pyrophosphate Detection Assay. *Analytical Biochemistry* **208**(1) (Jan. 1, 1993), 171–175. doi: 10.1006/abio.1993.1024 (cit. on p. 13).

45. E. R. Mardis: A Decade's Perspective on DNA Sequencing Technology. *Nature* **470**(7333) (7333 Feb. 2011), 198–203. doi: 10.1038/nature09796 (cit. on p. 14).

46. N. Stoler and A. Nekrutenko: Sequencing Error Profiles of Illumina Sequencing Instruments. *NAR Genomics and Bioinformatics* **3**(1) (Mar. 1, 2021), lqab019. doi: 10.1093/nargab/lqab019 (cit. on p. 14).

47. *Sequencing Technology | Sequencing by Synthesis*. Illumina. https://www.illumina.com/science/technology/next-generation-sequencing/sequencing-technology.html (cit. on p. 14).

48. M. O. Pollard, D. Gurdasani, A. J. Mentzer, T. Porter, and M. S. Sandhu: Long Reads: Their Purpose and Place. *Human Molecular Genetics* **27**(R2) (Aug. 1, 2018), R234–r241. doi: 10.1093/hmg/ddy177 (cit. on pp. 14, 40).

49. J. Eid et al.: Real-Time DNA Sequencing from Single Polymerase Molecules. *Science* **323**(5910) (Jan. 2, 2009), 133–138. doi: 10.1126/science.1162986 (cit. on p. 14).

50. M. J. Levene et al.: Zero-Mode Waveguides for Single-Molecule Analysis at High Concentrations. *Science* **299**(5607) (Jan. 31, 2003), 682–686. doi: 10.1126/science.1079700 (cit. on p. 14).

51. J. Clarke et al.: Continuous Base Identification for Single-Molecule Nanopore DNA Sequencing. *Nature Nanotechnology* **4**(4) (4 Apr. 2009), 265–270. doi: 10.1038/nnano.2009.12 (cit. on p. 15).

52. D. Deamer, M. Akeson, and D. Branton: Three Decades of Nanopore Sequencing. *Nature Biotechnology* **34**(5) (5 May 2016), 518–524. doi: 10.1038/nbt.3423 (cit. on p. 15).

53. R. R. Wick, L. M. Judd, and K. E. Holt: Performance of Neural Network Basecalling Tools for Oxford Nanopore Sequencing. *Genome Biology* **20**(1) (June 24, 2019), 129. doi: 10.1186/s13059-019-1727-y (cit. on pp. 15, 18, 20).

54. A. Rhoads and K. F. Au: PacBio Sequencing and Its Applications. *Genomics, Proteomics & Bioinformatics* **13**(5) (Oct. 1, 2015), 278–289. doi: 10.1016/j.gpb.2015.08.002 (cit. on p. 15).

55. C. L. Ip et al.: MinION Analysis and Reference Consortium: Phase 1 Data Release and Analysis. *F1000Research* **4** (Oct. 15, 2015), 1075. doi: 10.12688/f1000research.7201.1 (cit. on p. 15).

56. G. A. Logsdon, M. R. Vollger, and E. E. Eichler: Long-Read Human Genome Sequencing and Its Applications. *Nature Reviews Genetics* **21**(10) (10 Oct. 2020), 597–614. doi: 10.1038/s41576-020-0236-x (cit. on pp. 15–17, 19, 40).

57. M. Jain et al.: Nanopore Sequencing and Assembly of a Human Genome with Ultra-Long Reads. *Nature Biotechnology* **36**(4) (4 Apr. 2018), 338–345. doi: 10.1038/nbt.4060 (cit. on p. 15).

58. *Thar She Blows! Ultra Long Read Method for Nanopore Sequencing · Loman Labs*. Loman Labs. http://lab.loman.net/2017/03/09/ultrareads-for-nanopore/ (cit. on p. 15).

59. A. Payne, N. Holmes, V. Rakyan, and M. Loose: BulkVis: A Graphical Viewer for Oxford Nanopore Bulk FAST5 Files. *Bioinformatics* **35**(13) (July 1, 2019), 2193–2198. doi: 10.1093/bioinformatics/bty841 (cit. on p. 15).

60. V. Murigneux et al.: Comparison of Long-Read Methods for Sequencing and Assembly of a Plant Genome. *GigaScience* **9**(12) (Nov. 30, 2020), giaa146. doi: 10.1093/gigascience/giaa146 (cit. on p. 15).

61. M. J. P. Chaisson et al.: Resolving the Complexity of the Human Genome Using Single-Molecule Sequencing. *Nature* **517**(7536) (7536 Jan. 2015), 608–611. doi: 10.1038/nature13907 (cit. on p. 16).

62. M. Jain, H. E. Olsen, B. Paten, and M. Akeson: The Oxford Nanopore MinION: Delivery of Nanopore Sequencing to the Genomics Community. *Genome Biology* **17**(1) (Nov. 25, 2016), 239. doi: 10.1186/s13059-016-1103-0 (cit. on p. 16).

63. M. Hong et al.: RNA Sequencing: New Technologies and Applications in Cancer Research. *Journal of Hematology & Oncology* **13**(1) (Dec. 4, 2020), 166. doi: 10.1186/s13045-020-01005-x (cit. on p. 16).

64. F. Ozsolak and P. M. Milos: RNA Sequencing: Advances, Challenges and Opportunities. *Nature Reviews Genetics* **12**(2) (2 Feb. 2011), 87–98. doi: 10.1038/nrg2934 (cit. on p. 16).

65. D. F. Hunt, J. R. Yates, J. Shabanowitz, S. Winston, and C. R. Hauer: Protein Sequencing by Tandem Mass Spectrometry. *Proceedings of the National Academy of Sciences* **83**(17) (Sept. 1986), 6233–6237. doi: 10.1073/pnas.83.17.6233 (cit. on p. 16).

66. B. J. Smith: *Protein Sequencing Protocols*. 2002. doi: 10.1385/1592593429 (cit. on p. 16).

67. L. Restrepo-Pérez, C. Joo, and C. Dekker: Paving the Way to Single-Molecule Protein Sequencing. *Nature Nanotechnology* **13**(9) (9 Sept. 2018), 786–796. doi: 10.1038/s41565-018-0236-6 (cit. on p. 16).

68. J. L. Weirather et al.: Comprehensive Comparison of Pacific Biosciences and Oxford Nanopore Technologies and Their Applications to Transcriptome Analysis. *F1000Research* **6** (June 19, 2017), 100. doi: 10.12688/f1000research.10571.2 (cit. on pp. 16, 19).

69. Y. Wang, Y. Zhao, A. Bollas, Y. Wang, and K. F. Au: Nanopore Sequencing Technology, Bioinformatics and Applications. *Nature Biotechnology* **39**(11) (11 Nov. 2021), 1348–1365. doi: 10.1038/s41587-021-01108-x (cit. on p. 16).

70. X. Ma et al.: Analysis of Error Profiles in Deep Next-Generation Sequencing Data. *Genome Biology* **20**(1) (Mar. 14, 2019), 50. doi: 10.1186/s13059-019-1659-6 (cit. on p. 16).

71. L. Lima et al.: Comparative Assessment of Long-Read Error Correction Software Applied to Nanopore RNA-sequencing Data. *Briefings in Bioinformatics* **21**(4) (July 15, 2020), 1164–1181. doi: 10.1093/bib/bbz058 (cit. on p. 17).

72. S. Fu, A. Wang, and K. F. Au: A Comparative Evaluation of Hybrid Error Correction Methods for Error-Prone Long Reads. *Genome Biology* **20**(1) (Feb. 4, 2019), 26. doi: 10.1186/s13059-018-1605-z (cit. on p. 17).

73. H. Zhang, C. Jain, and S. Aluru: A Comprehensive Evaluation of Long Read Error Correction Methods. *BMC Genomics* **21**(6) (Dec. 21, 2020), 889. doi: 10.1186/s12864-020-07227-0 (cit. on p. 17).

74. S. L. Amarasinghe et al.: Opportunities and Challenges in Long-Read Sequencing Data Analysis. *Genome Biology* **21**(1) (Feb. 7, 2020), 30. doi: 10.1186/s13059-020-1935-5 (cit. on pp. 17, 18, 21).

75. J. Ruan and H. Li: Fast and Accurate Long-Read Assembly with Wtdbg2. *Nature Methods* **17**(2) (2 Feb. 2020), 155–158. doi: 10.1038/s41592-019-0669-3 (cit. on pp. 17, 21).

76. S. Koren et al.: Canu: Scalable and Accurate Long-Read Assembly via Adaptive k-Mer Weighting and Repeat Separation. *Genome Research* **27**(5) (Jan. 5, 2017), 722–736. doi: 10.1101/gr.215087.116 (cit. on p. 17).

77. G. Tischler and E. W. Myers: *Non Hybrid Long Read Consensus Using Local De Bruijn Graph Assembly*. Feb. 6, 2017. doi: 10.1101/106252 (cit. on p. 17).

78. R. L. Warren et al.: ntEdit: Scalable Genome Sequence Polishing. *Bioinformatics* **35**(21) (Nov. 1, 2019), 4430–4432. doi: 10.1093/bioinformatics/btz400 (cit. on p. 17).

79. N. L. Hepler et al.: An Improved Circular Consensus Algorithm with an Application to Detect HIV-1 Drug-Resistance Associated Mutations (DRAMs). *Conference on Advances in Genome Biology and Technology*. 2016, 1. https://www.pacb.com/wp-content/uploads/improved-circular-consensus-algorithm-with-application-to-detection-hiv-1-drams.pdf (cit. on p. 17).

80. J. T. Simpson et al.: Detecting DNA Cytosine Methylation Using Nanopore Sequencing. *Nature Methods* **14**(4) (4 Apr. 2017), 407–410. doi: 10.1038/nmeth.4184 (cit. on pp. 17, 20).

81. R. Vaser, I. Sović, N. Nagarajan, and M. Šikić: Fast and Accurate de Novo Genome Assembly from Long Uncorrected Reads. *Genome Research* **27**(5) (May 2017), 737–746. doi: 10.1101/gr.214270.116 (cit. on p. 17).

82. T. Hackl, R. Hedrich, J. Schultz, and F. Förster: Proovread : Large-Scale High-Accuracy PacBio Correction through Iterative Short Read Consensus. *Bioinformatics* **30**(21) (Nov. 1, 2014), 3004–3011. doi: 10.1093/bioinformatics/btu392 (cit. on pp. 17, 24).

83. G. Miclotte et al.: Jabba: Hybrid Error Correction for Long Sequencing Reads. *Algorithms for Molecular Biology* **11**(1) (May 3, 2016), 10. doi: 10.1186/s13015-016-0075-7 (cit. on p. 17).

84. S. Koren et al.: Hybrid Error Correction and de Novo Assembly of Single-Molecule Sequencing Reads. *Nature Biotechnology* **30**(7) (7 July 2012), 693–700. doi: 10.1038/nbt.2280 (cit. on pp. 17, 24).

85. L. Salmela and E. Rivals: LoRDEC: Accurate and Efficient Long Read Error Correction. *Bioinformatics* **30**(24) (Dec. 15, 2014), 3506–3514. doi: 10.1093/bioinformatics/btu538 (cit. on p. 17).

86. B. J. Walker et al.: Pilon: An Integrated Tool for Comprehensive Microbial Variant Detection and Genome Assembly Improvement. *Plos One* **9**(11) (Nov. 19, 2014), e112963. doi: 10.1371/journal.pone.0112963 (cit. on pp. 17, 20).

87. A. M. Wenger et al.: Accurate Circular Consensus Long-Read Sequencing Improves Variant Detection and Assembly of a Human Genome. *Nature Biotechnology* **37**(10) (Oct. 2019), 1155–1162. doi: 10.1038/s41587-019-0217-9 (cit. on pp. 17, 19).

88. W. Timp, J. Comer, and A. Aksimentiev: DNA Base-Calling from a Nanopore Using a Viterbi Algorithm. *Biophysical Journal* **102**(10) (May 16, 2012), L37–l39. doi: 10.1016/j.bpj.2012.04.009 (cit. on p. 18).

89. P. Perešíni, V. Boža, B. Brejová, and T. Vinař: Nanopore Base Calling on the Edge. *Bioinformatics* **37**(24) (Dec. 15, 2021), 4661–4667. doi: 10.1093/bioinformatics/btab528 (cit. on p. 18).

90. V. Boža, B. Brejová, and T. Vinař: DeepNano: Deep Recurrent Neural Networks for Base Calling in MinION Nanopore Reads. *Plos One* **12**(6) (June 5, 2017), e0178751. doi: 10.1371/journal.pone.0178751 (cit. on p. 18).

91. A. D. Tyler et al.: Evaluation of Oxford Nanopore's MinION Sequencing Device for Microbial Whole Genome Sequencing Applications. *Scientific Reports* **8**(1) (1 July 19, 2018), 10931. doi: 10.1038/s41598-018-29334-5 (cit. on p. 18).

92. B. Lin, J. Hui, and H. Mao: Nanopore Technology and Its Applications in Gene Sequencing. *Biosensors* **11**(7) (7 July 2021), 214. doi: 10.3390/bios11070214 (cit. on p. 18).

93. *Oxford Nanopore Tech Update: New Duplex Method for Q30 Nanopore Single Molecule Reads, PromethION 2, and More*. Oxford Nanopore Technologies. http://nanoporetech.com/about-us/news/oxford-nanopore-tech-update-new-duplex-method-q30-nanopore-single-molecule-reads-0 (cit. on p. 18).

94. N. Sanderson et al.: *Comparison of R9.4.1/Kit10 and R10/Kit12 Oxford Nanopore Flowcells and Chemistries in Bacterial Genome Reconstruction*. June 13, 2022. doi: 10.1101/2022.04.29.490057 (cit. on p. 18).

95. S. M. Karst et al.: High-Accuracy Long-Read Amplicon Sequences Using Unique Molecular Identifiers with Nanopore or PacBio Sequencing. *Nature Methods* **18**(2) (2 Feb. 2021), 165–169. doi: 10.1038/s41592-020-01041-y (cit. on p. 18).

96. Z. Chen et al.: Highly Accurate Fluorogenic DNA Sequencing with Information Theory–Based Error Correction. *Nature Biotechnology* **35**(12) (12 Dec. 2017), 1170–1178. doi: 10.1038/nbt.3982 (cit. on p. 18).

97. *High Performance Long Read Assay Enables Contiguous Data up to 10Kb on Existing Illumina Platforms*. Illumina. https://www.illumina.com/content/illumina-marketing/amr/en_US/science/genomics-research/articles/infinity-high-performance-long-read-assay.html (cit. on pp. 18, 21).

98. A. S. Booeshaghi and L. Pachter: *Pseudoalignment Facilitates Assignment of Error-Prone Ultima Genomics Reads*. June 5, 2022. doi: 10.1101/2022.06.04.494845 (cit. on p. 19).

99. C. Delahaye and J. Nicolas: Sequencing DNA with Nanopores: Troubles and Biases. *Plos One* **16**(10) (Oct. 1, 2021), e0257521. doi: 10.1371/journal.pone.0257521 (cit. on p. 19).

100. S. Goodwin et al.: Oxford Nanopore Sequencing, Hybrid Error Correction, and de Novo Assembly of a Eukaryotic Genome. *Genome Research* **25**(11) (Jan. 11, 2015), 1750–1756. doi: 10.1101/gr.191395.115 (cit. on p. 19).

101. J. C. Dohm, P. Peters, N. Stralis-Pavese, and H. Himmelbauer: Benchmarking of Long-Read Correction Methods. *NAR Genomics and Bioinformatics* **2**(2) (June 1, 2020). doi: 10.1093/nargab/lqaa037 (cit. on pp. 19, 47).

102. J. Foox et al.: Performance Assessment of DNA Sequencing Platforms in the ABRF Next-Generation Sequencing Study. *Nature Biotechnology* **39**(9) (9 Sept. 2021), 1129–1140. doi: 10.1038/s41587-021-01049-5 (cit. on p. 20).

103. Y.-T. Huang, P.-Y. Liu, and P.-W. Shih: Homopolish: A Method for the Removal of Systematic Errors in Nanopore Sequencing by Homologous Polishing. *Genome Biology* **22**(1) (Mar. 31, 2021), 95. doi: 10.1186/s13059-021-02282-6 (cit. on p. 20).

104. F. J. Rang, W. P. Kloosterman, and J. de Ridder: From Squiggle to Basepair: Computational Approaches for Improving Nanopore Sequencing Read Accuracy. *Genome Biology* **19**(1) (July 13, 2018), 90. doi: 10.1186/s13059-018-1462-9 (cit. on p. 20).

105. P. Sarkozy, Á. Jobbágy, and P. Antal: Calling Homopolymer Stretches from Raw Nanopore Reads by Analyzing K-Mer Dwell Times. *Embec & Nbc 2017*. Singapore, 2018, 241–244. doi: 10.1007/978-981-10-5122-7_61 (cit. on p. 20).

106. J. A. Hawkins, S. K. Jones, I. J. Finkelstein, and W. H. Press: Indel-Correcting DNA Barcodes for High-Throughput Sequencing. *Proceedings of the National Academy of Sciences* **115**(27) (July 3, 2018), E6217–e6226. doi: 10.1073/pnas.1802640115 (cit. on p. 20).

107. A. Srivathsan et al.: A MinION™-Based Pipeline for Fast and Cost-Effective DNA Barcoding. *Molecular Ecology Resources* **18**(5) (2018), 1035–1049. doi: 10.1111/1755-0998.12890 (cit. on p. 20).

108. Y. Wang, M. Noor-A-Rahim, E. Gunawan, Y. L. Guan, and C. L. Poh: Construction of Bio-Constrained Code for DNA Data Storage. *IEEE Communications Letters* **23**(6) (June 2019), 963–966. doi: 10.1109/lcomm.2019.2912572 (cit. on p. 20).

109. *R10.3: The Newest Nanopore for High Accuracy Nanopore Sequencing – Now Available in Store*. Oxford Nanopore Technologies. http://nanoporetech.com/about-us/news/r103-newest-nanopore-high-accuracy-nanopore-sequencing-now-available-store (cit. on p. 21).

110. L. Zhou et al.: Detection of DNA Homopolymer with Graphene Nanopore. *Journal of Vacuum Science & Technology B* **37**(6) (Nov. 2019), 061809. doi: 10.1116/1.5116295 (cit. on p. 21).

111. Y. Goto, I. Yanagi, K. Matsui, T. Yokoi, and K.-i. Takeda: Identification of Four Single-Stranded DNA Homopolymers with a Solid-State Nanopore in Alkaline CsCl Solution. *Nanoscale* **10**(44) (2018), 20844–20850. doi: 10.1039/c8nr04238a (cit. on p. 21).

112. S. Nurk et al.: HiCanu: Accurate Assembly of Segmental Duplications, Satellites, and Allelic Variants from High-Fidelity Long Reads. *Genome Research* **30**(9) (Jan. 9, 2020), 1291–1305. doi: 10.1101/gr.263566.120 (cit. on pp. 21, 47).

113. B. Ekim, B. Berger, and R. Chikhi: Minimizer-Space de Bruijn Graphs: Whole-genome Assembly of Long Reads in Minutes on a Personal Computer. *Cell Systems* **12**(10) (Oct. 20, 2021), 958–968.e6. doi: 10.1016/j.cels.2021.08.009 (cit. on pp. 21, 47).

114. K. Shafin et al.: Nanopore Sequencing and the Shasta Toolkit Enable Efficient de Novo Assembly of Eleven Human Genomes. *Nature Biotechnology* **38**(9) (9 Sept. 2020), 1044–1053. doi: 10.1038/s41587-020-0503-6 (cit. on p. 21).

115. J. R. Miller et al.: Aggressive Assembly of Pyrosequencing Reads with Mates. *Bioinformatics* **24**(24) (Dec. 15, 2008), 2818–2824. doi: 10.1093/bioinformatics/btn548 (cit. on pp. 21, 47).

116. K. Sahlin and P. Medvedev: De Novo Clustering of Long-Read Transcriptome Data Using a Greedy, Quality Value-Based Algorithm. *Journal of Computational Biology* **27**(4) (Apr. 1, 2020), 472–484. doi: 10.1089/cmb.2019.0299 (cit. on pp. 21, 47, 67).

117. K. F. Au, J. G. Underwood, L. Lee, and W. H. Wong: Improving PacBio Long Read Accuracy by Short Read Alignment. *Plos One* **7**(10) (Oct. 4, 2012), e46679. doi: 10.1371/journal.pone.0046679 (cit. on pp. 21, 47).

118. R. Hu, G. Sun, and X. Sun: LSCplus: A Fast Solution for Improving Long Read Accuracy by Short Read Alignment. *BMC Bioinformatics* **17**(1) (Nov. 9, 2016), 451. doi: 10.1186/s12859-016-1316-y (cit. on p. 21).

119. H. Li: Minimap2: Pairwise Alignment for Nucleotide Sequences. *Bioinformatics* **34**(18) (Sept. 15, 2018), 3094–3100. doi: 10.1093/bioinformatics/bty191 (cit. on pp. 21, 36, 39, 40, 47, 55, 64, 65, 145).

120. C. Jain et al.: Weighted Minimizer Sampling Improves Long Read Mapping. *Bioinformatics* **36** (Supplement_1 July 1, 2020), i111–i118. doi: 10.1093/bioinformatics/btaa435 (cit. on pp. 21, 36, 40, 58, 145).

121. C. Van Neste, F. Van Nieuwerburgh, D. Van Hoofstat, and D. Deforce: Forensic STR Analysis Using Massive Parallel Sequencing. *Forensic Science International: Genetics* **6**(6) (Dec. 1, 2012), 810–818. doi: 10.1016/j.fsigen.2012.03.004 (cit. on p. 21).

122. *Short-Read Sequencing by Binding*. PacBio. https://www.pacb.com/technology/sequencing-by-binding/ (cit. on p. 21).

123. A. E. Cetin et al.: Plasmonic Sensor Could Enable Label-Free DNA Sequencing. *ACS Sensors* **3**(3) (Mar. 23, 2018), 561–568. doi: 10.1021/acssensors.7b00957 (cit. on p. 21).

124. G. Almogy et al.: *Cost-Efficient Whole Genome-Sequencing Using Novel Mostly Natural Sequencing-by-Synthesis Chemistry and Open Fluidics Platform*. June 3, 2022. doi: 10.1101/2022.05.29.493900 (cit. on p. 22).

125. S. Sunagawa et al.: Tara Oceans: Towards Global Ocean Ecosystems Biology. *Nature Reviews Microbiology* **18**(8) (8 Aug. 2020), 428–445. doi: 10.1038/s41579-020-0364-5 (cit. on p. 22).

126. H. A. Lewin et al.: Earth BioGenome Project: Sequencing Life for the Future of Life. *Proceedings of the National Academy of Sciences* **115**(17) (Apr. 24, 2018), 4325–4333. doi: 10.1073/pnas.1720115115 (cit. on p. 22).

127. G. Lightbody et al.: Review of Applications of High-Throughput Sequencing in Personalized Medicine: Barriers and Facilitators of Future Progress in Research and Clinical Application. *Briefings in Bioinformatics* **20**(5) (Sept. 27, 2019), 1795–1811. doi: 10.1093/bib/bby051 (cit. on p. 22).

128. R. W. Hamming: *Coding and Information Theory*. 1980. isbn: 978-0-13-139139-0 (cit. on p. 23).

129. D. Gusfield: *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge, 1997. doi: 10.1017/cbo9780511574931 (cit. on pp. 23, 37–39, 47).

130. V. I. Levenshtein: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* **10** (Feb. 1, 1966), 707. https://ui.adsabs.harvard.edu/abs/1966SPhD...10..707L (cit. on p. 23).

131. R. C. Hardison: Comparative Genomics. *PLOS Biology* **1**(2) (Nov. 17, 2003), e58. doi: 10.1371/journal.pbio.0000058 (cit. on p. 24).

132. J. Felsenstein: Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach. *Journal of Molecular Evolution* **17**(6) (Nov. 1, 1981), 368–376. doi: 10.1007/bf01734359 (cit. on p. 24).

133. S. Kumar, K. Tamura, and M. Nei: MEGA: Molecular Evolutionary Genetics Analysis Software for Microcomputers. *Bioinformatics* **10**(2) (Apr. 2, 1994), 189–191. doi: 10.1093/bioinformatics/10.2.189 (cit. on p. 24).

134. A. M. Kozlov, D. Darriba, T. Flouri, B. Morel, and A. Stamatakis: RAxML-NG: A Fast, Scalable and User-Friendly Tool for Maximum Likelihood Phylogenetic Inference. *Bioinformatics* **35**(21) (Nov. 1, 2019), 4453–4455. doi: 10.1093/bioinformatics/btz305 (cit. on p. 24).

135. S. Guindon et al.: New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. *Systematic Biology* **59**(3) (May 1, 2010), 307–321. doi: 10.1093/sysbio/syq010 (cit. on p. 24).

136. M. N. Price, P. S. Dehal, and A. P. Arkin: FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments. *Plos One* **5**(3) (Mar. 10, 2010), e9490. doi: 10.1371/journal.pone.0009490 (cit. on p. 24).

137. J. Jumper et al.: Highly Accurate Protein Structure Prediction with AlphaFold. *Nature* **596**(7873) (7873 Aug. 2021), 583–589. doi: 10.1038/s41586-021-03819-2 (cit. on pp. 24, 68, 130, 132).

138. K. Karplus et al.: Predicting Protein Structure Using Only Sequence Information. *Proteins: Structure, Function, and Bioinformatics* **37**(S3) (1999), 121–125. doi: 10.1002/(sici)1097-0134(1999)37:3+<121::aid-prot16>3.0.co;2-q (cit. on p. 24).

139. J. D. Watson, R. A. Laskowski, and J. M. Thornton: Predicting Protein Function from Sequence and Structural Data. *Current Opinion in Structural Biology* **15**(3) (June 1, 2005), 275–284. doi: 10.1016/j.sbi.2005.04.003 (cit. on p. 24).

140. D. Lee, O. Redfern, and C. Orengo: Predicting Protein Function from Sequence and Structure. *Nature Reviews Molecular Cell Biology* **8**(12) (12 Dec. 2007), 995–1005. doi: 10.1038/nrm2281 (cit. on p. 24).

141. L. Salmela and J. Schröder: Correcting Errors in Short Reads by Multiple Alignments. *Bioinformatics* **27**(11) (June 1, 2011), 1455–1461. doi: 10.1093/bioinformatics/btr170 (cit. on p. 24).

142. P. Medvedev, M. Stanciu, and M. Brudno: Computational Methods for Discovering Structural Variation with Next-Generation Sequencing. *Nature Methods* **6**(11) (11 Nov. 2009), S13–s20. doi: 10.1038/nmeth.1374 (cit. on p. 24).

143. M. Mahmoud et al.: Structural Variant Calling: The Long and the Short of It. *Genome Biology* **20**(1) (Nov. 20, 2019), 246. doi: 10.1186/s13059-019-1828-7 (cit. on p. 24).

144. W.-K. Sung: *Algorithms in Bioinformatics: A Practical Introduction*. New York, Oct. 10, 2011. doi: 10.1201/9781420070347 (cit. on pp. 24, 27, 28, 42).

145. S. B. Needleman and C. D. Wunsch: A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology* **48**(3) (Mar. 28, 1970), 443–453. doi: 10.1016/0022-2836(70)90057-4 (cit. on p. 24).

146. T. F. Smith and M. S. Waterman: Identification of Common Molecular Subsequences. *Journal of Molecular Biology* **147**(1) (Mar. 25, 1981), 195–197. doi: 10.1016/0022-2836(81)90087-5 (cit. on pp. 24, 27).

147. S. P. Bradley, A. C. Hax, and T. L. Magnanti: *Applied Mathematical Programming*. 1977. https://web.mit.edu/15.053/www/AMP.htm (cit. on p. 24).

148. R. Bellman: The Theory of Dynamic Programming. *Bulletin of the American Mathematical Society* **60**(6) (1954), 503–515. doi: 10.1090/s0002-9904-1954-09848-8 (cit. on p. 24).

149. W. J. Masek and M. S. Paterson: A Faster Algorithm Computing String Edit Distances. *Journal of Computer and System Sciences* **20**(1) (Feb. 1, 1980), 18–31. doi: 10.1016/0022-0000(80)90002-1 (cit. on p. 27).

150. N. X. Vinh, J. Epps, and J. Bailey: Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. en. *Journal of Machine Learning Research* **11** (2010), 18. https://jmlr.org/papers/volume11/vinh10a/vinh10a.pdf (cit. on pp. 27, 113, 159).

151. J. D. Ullman, A. V. Aho, and D. S. Hirschberg: Bounds on the Complexity of the Longest Common Subsequence Problem. *Journal of the ACM* **23**(1) (Jan. 1, 1976), 1–12. doi: 10.1145/321921.321922 (cit. on p. 27).

152. D. S. Hirschberg: A Linear Space Algorithm for Computing Maximal Common Subsequences. *Communications of the ACM* **18**(6) (June 1, 1975), 341–343. doi: 10.1145/360825.360861 (cit. on p. 27).

153. E. W. Myers and W. Miller: Optimal Alignments in Linear Space. *Bioinformatics* **4**(1) (Mar. 1, 1988), 11–17. doi: 10.1093/bioinformatics/4.1.11 (cit. on p. 27).

154. P. Rice, I. Longden, and A. Bleasby: EMBOSS: The European Molecular Biology Open Software Suite. *Trends in genetics* **16**(6) (2000), 276–277. doi: 10.1016/s0168-9525(00)02024-2 (cit. on pp. 27, 28).

155. X. Huang and W. Miller: A Time-Efficient, Linear-Space Local Similarity Algorithm. *Advances in Applied Mathematics* **12**(3) (Sept. 1, 1991), 337–357. doi: 10.1016/0196-8858(91)90017-d (cit. on p. 28).

156. M. S. Waterman and M. Eggert: A New Algorithm for Best Subsequence Alignments with Application to tRNA-rRNA Comparisons. *Journal of Molecular Biology* **197**(4) (Oct. 20, 1987), 723–728. doi: 10.1016/0022-2836(87)90478-5 (cit. on p. 28).

157. J. E. Stajich et al.: The Bioperl Toolkit: Perl Modules for the Life Sciences. *Genome Research* **12**(10) (Jan. 10, 2002), 1611–1618. doi: 10.1101/gr.361602 (cit. on p. 28).

158. R. C. Gentleman et al.: Bioconductor: Open Software Development for Computational Biology and Bioinformatics. *Genome Biology* **5**(10) (Sept. 15, 2004), R80. doi: 10.1186/gb-2004-5-10-r80 (cit. on p. 28).

159. J. Daily: Parasail: SIMD C Library for Global, Semi-Global, and Local Pairwise Sequence Alignments. *BMC Bioinformatics* **17**(1) (Feb. 10, 2016), 81. doi: 10.1186/s12859-016-0930-z (cit. on p. 28).

160. W. Frohmberg, M. Kierzynka, J. Blazewicz, and P. Wojciechowski: G-PAS 2.0 – an Improved Version of Protein Alignment Tool with an Efficient Backtracking Routine on Multiple GPUs. *Bulletin of the Polish Academy of Sciences: Technical Sciences* **60**(3) (Dec. 1, 2012), 491–494. doi: 10.2478/v10175-012-0062-1 (cit. on p. 28).

161. S. F. Altschul: Substitution Matrices. *eLS*. 2013. doi: 10.1002/9780470015902.a0005265.pub3 (cit. on pp. 29, 30).

162. M. Dayhoff, R. Schwartz, and B. Orcutt: A Model of Evolutionary Change in Proteins. *Atlas of Protein Sequence and Structure*. 1978, pp. 345–352 (cit. on p. 30).

163. T. Müller and M. Vingron: Modeling Amino Acid Replacement. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology* **7**(6) (2000), 761–776. doi: 10.1089/10665270050514918 (cit. on p. 30).

164. S. Henikoff and J. G. Henikoff: Amino Acid Substitution Matrices from Protein Blocks. *Proceedings of the National Academy of Sciences* **89**(22) (Nov. 15, 1992), 10915–10919. doi: 10.1073/pnas.89.22.10915 (cit. on p. 31).

165. S. Whelan and N. Goldman: A General Empirical Model of Protein Evolution Derived from Multiple Protein Families Using a Maximum-Likelihood Approach. *Molecular Biology and Evolution* **18**(5) (May 2001), 691–699. doi: 10.1093/oxfordjournals.molbev.a003851 (cit. on p. 31).

166. S. Q. Le and O. Gascuel: An Improved General Amino Acid Replacement Matrix. *Molecular Biology and Evolution* **25**(7) (July 1, 2008), 1307–1320. doi: 10.1093/molbev/msn067 (cit. on p. 31).

167. T. Müller, S. Rahmann, and M. Rehmsmeier: Non-Symmetric Score Matrices and the Detection of Homologous Transmembrane Proteins. *Bioinformatics* **17** (suppl_1 June 1, 2001), S182–s189. doi: 10.1093/bioinformatics/17.suppl_1.S182 (cit. on p. 31).

168. P. C. Ng, J. G. Henikoff, and S. Henikoff: PHAT: A Transmembrane-Specific Substitution Matrix. Predicted Hydrophobic and Transmembrane. *Bioinformatics (Oxford, England)* **16**(9) (Sept. 2000), 760–766. doi: 10.1093/bioinformatics/16.9.760 (cit. on p. 31).

169. R. Trivedi and H. A. Nagarajaram: Amino Acid Substitution Scoring Matrices Specific to Intrinsically Disordered Regions in Proteins. *Scientific Reports* **9**(1) (1 Nov. 8, 2019), 16380. doi: 10.1038/s41598-019-52532-8 (cit. on p. 31).

170. N. C. W. Goonesekere and B. Lee: Context-Specific Amino Acid Substitution Matrices and Their Use in the Detection of Protein Homologs. *Proteins: Structure, Function, and Bioinformatics* **71**(2) (2008), 910–919. doi: 10.1002/prot.21775 (cit. on p. 31).

171. U. Paila, R. Kondam, and A. Ranjan: Genome Bias Influences Amino Acid Choices: Analysis of Amino Acid Substitution and Re-Compilation of Substitution Matrices Exclusive to an AT-biased Genome. *Nucleic Acids Research* **36**(21) (Dec. 2008), 6664–6675. doi: 10.1093/nar/gkn635 (cit. on p. 31).

172. D. C. Nickle et al.: HIV-Specific Probabilistic Models of Protein Evolution. *PLoS ONE* **2**(6) (June 6, 2007), e503. doi: 10.1371/journal.pone.0000503 (cit. on p. 31).

173. M. E. Sardiu, G. Alves, and Y.-K. Yu: Score Statistics of Global Sequence Alignment from the Energy Distribution of a Modified Directed Polymer and Directed Percolation Problem. *Physical Review. E* **72** (6 Pt 1 Dec. 2005), 061917. doi: 10.1103/PhysRevE.72.061917 (cit. on p. 31).

174. F. Chiaromonte, V. B. Yap, and W. Miller: Scoring Pairwise Genomic Sequence Alignments. *Biocomputing 2002*. Kauai, Hawaii, USA, Dec. 2001, 115–126. doi: 10.1142/9789812799623_0012 (cit. on p. 31).

175. A. Schneider, G. M. Cannarozzi, and G. H. Gonnet: Empirical Codon Substitution Matrix. *BMC bioinformatics* **6** (June 1, 2005), 134. doi: 10.1186/1471-2105-6-134 (cit. on p. 31).

176. A. Doron-Faigenboim and T. Pupko: A Combined Empirical and Mechanistic Codon Model. *Molecular Biology and Evolution* **24**(2) (Feb. 1, 2007), 388–397. doi: 10.1093/molbev/msl175 (cit. on p. 31).

177. R. A. Cartwright: Problems and Solutions for Estimating Indel Rates and Length Distributions. *Molecular Biology and Evolution* **26**(2) (Feb. 1, 2009), 473–480. doi: 10.1093/molbev/msn275 (cit. on p. 31).

178. W. M. Fitch and T. F. Smith: Optimal Sequence Alignments. *Proceedings of the National Academy of Sciences* **80**(5) (Mar. 1983), 1382–1386. doi: 10.1073/pnas.80.5.1382 (cit. on p. 32).

179. M. S. Waterman, T. F. Smith, and W. A. Beyer: Some Biological Sequence Metrics. *Advances in Mathematics* **20**(3) (June 1, 1976), 367–387. doi: 10.1016/0001-8708(76)90202-4 (cit. on p. 32).

180. O. Gotoh: An Improved Algorithm for Matching Biological Sequences. *Journal of Molecular Biology* **162**(3) (Dec. 15, 1982), 705–708. doi: 10.1016/0022-2836(82)90398-9 (cit. on p. 32).

181. S. F. Altschul and B. W. Erickson: Optimal Sequence Alignment Using Affine Gap Costs. *Bulletin of Mathematical Biology* **48**(5) (Jan. 1, 1986), 603–616. doi: 10.1016/s0092-8240(86)90010-8 (cit. on p. 32).

182. M. S. Waterman: Efficient Sequence Alignment Algorithms. *Journal of Theoretical Biology* **108**(3) (June 7, 1984), 333–337. doi: 10.1016/s0022-5193(84)80037-5 (cit. on p. 32).

183. W. Miller and E. W. Myers: Sequence Comparison with Concave Weighting Functions. *Bulletin of Mathematical Biology* **50**(2) (Mar. 1, 1988), 97–120. doi: 10.1007/bf02459948 (cit. on p. 32).

184. R. A. Cartwright: Logarithmic Gap Costs Decrease Alignment Accuracy. *BMC Bioinformatics* **7** (Dec. 5, 2006), 527. doi: 10.1186/1471-2105-7-527 (cit. on p. 32).

185. N. C. W. Goonesekere and B. Lee: Frequency of Gaps Observed in a Structurally Aligned Protein Pair Database Suggests a Simple Gap Penalty Function. *Nucleic Acids Research* **32**(9) (May 1, 2004), 2838–2843. doi: 10.1093/nar/gkh610 (cit. on p. 32).

186. S. A. Benner, M. A. Cohen, and G. H. Gonnet: Empirical and Structural Models for Insertions and Deletions in the Divergent Evolution of Proteins. *Journal of Molecular Biology* **229**(4) (Feb. 20, 1993), 1065–1082. doi: 10.1006/jmbi.1993.1105 (cit. on p. 32).

187. J. O. Wrabl and N. V. Grishin: Gaps in Structurally Similar Proteins: Towards Improvement of Multiple Sequence Alignment. *Proteins: Structure, Function, and Bioinformatics* **54**(1) (2004), 71–87. doi: 10.1002/prot.10508 (cit. on p. 32).

188. W. Zhang, S. Liu, and Y. Zhou: SP5: Improving Protein Fold Recognition by Using Torsion Angle Profiles and Profile-Based Gap Penalty Model. *Plos One* **3**(6) (June 4, 2008), e2325. doi: 10.1371/journal.pone.0002325 (cit. on p. 32).

189. F. Jeanmougin, J. D. Thompson, M. Gouy, D. G. Higgins, and T. J. Gibson: Multiple Sequence Alignment with Clustal X. *Trends in Biochemical Sciences* **23**(10) (Oct. 1, 1998), 403–405. doi: 10.1016/s0968-0004(98)01285-7 (cit. on p. 32).

190. C. Wang, R.-X. Yan, X.-F. Wang, J.-N. Si, and Z. Zhang: Comparison of Linear Gap Penalties and Profile-Based Variable Gap Penalties in Profile–Profile Alignments. *Computational Biology and Chemistry* **35**(5) (Oct. 12, 2011), 308–318. doi: 10.1016/j.compbiolchem.2011.07.006 (cit. on p. 32).

191. S. Marco-Sola, J. C. Moure, M. Moreto, and A. Espinosa: Fast Gap-Affine Pairwise Alignment Using the Wavefront Algorithm. *Bioinformatics* (btaa777 Sept. 11, 2020). doi: 10.1093/bioinformatics/btaa777 (cit. on p. 32).

192. W. R. Pearson and W. Miller: [27] Dynamic Programming Algorithms for Biological Sequence Comparison. *Methods in Enzymology*. **210**. Jan. 1, 1992, 575–601. doi: 10.1016/0076-6879(92)10029-d (cit. on p. 32).

193. J. L. Spouge: Speeding up Dynamic Programming Algorithms for Finding Optimal Lattice Paths. *SIAM Journal on Applied Mathematics* **49**(5) (Oct. 1989), 1552–1566. doi: 10.1137/0149094 (cit. on p. 33).

194. J. W. Fickett: Fast Optimal Alignment. *Nucleic Acids Research* **12** (1Part1 Jan. 11, 1984), 175–179. doi: 10.1093/nar/12.1Part1.175 (cit. on p. 33).

195. J. Chao, F. Tang, and L. Xu: Developments in Algorithms for Sequence Alignment: A Review. *Biomolecules* **12**(4) (Apr. 6, 2022), 546. doi: 10.3390/biom12040546 (cit. on pp. 33, 34, 41).

196. K. Katoh, K. Misawa, K.-i. Kuma, and T. Miyata: MAFFT: A Novel Method for Rapid Multiple Sequence Alignment Based on Fast Fourier Transform. *Nucleic Acids Research* **30**(14) (July 15, 2002), 3059–3066. doi: 10.1093/nar/gkf436 (cit. on pp. 33, 42, 43).

197. Y. Sun and J. Buhler: Choosing the Best Heuristic for Seeded Alignment of DNA Sequences. *BMC Bioinformatics* **7**(1) (Mar. 13, 2006), 133. doi: 10.1186/1471-2105-7-133 (cit. on p. 35).

198. H. Li and N. Homer: A Survey of Sequence Alignment Algorithms for Next-Generation Sequencing. *Briefings in Bioinformatics* **11**(5) (Sept. 1, 2010), 473–483. doi: 10.1093/bib/bbq015 (cit. on p. 35).

199. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman: Basic Local Alignment Search Tool. *Journal of Molecular Biology* **215**(3) (Oct. 5, 1990), 403–410. doi: 10.1016/s0022-2836(05)80360-2 (cit. on p. 35).

200. S. F. Altschul et al.: Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs. *Nucleic Acids Research* **25**(17) (Sept. 1, 1997), 3389–3402. doi: 10.1093/nar/25.17.3389 (cit. on pp. 35, 133).

201. S. Schwartz et al.: Human–Mouse Alignments with BLASTZ. *Genome Research* **13**(1) (Jan. 1, 2003), 103–107. doi: 10.1101/gr.809403 (cit. on p. 35).

202. B. Ma, J. Tromp, and M. Li: PatternHunter: Faster and More Sensitive Homology Search. *Bioinformatics* **18**(3) (Mar. 1, 2002), 440–445. doi: 10.1093/bioinformatics/18.3.440 (cit. on p. 35).

203. R. C. Edgar: Search and Clustering Orders of Magnitude Faster than BLAST. *Bioinformatics* **26**(19) (Oct. 1, 2010), 2460–2461. doi: 10.1093/bioinformatics/btq461 (cit. on p. 35).

204. B. Buchfink, C. Xie, and D. H. Huson: Fast and Sensitive Protein Alignment Using DIAMOND. *Nature Methods* **12**(1) (Jan. 2015), 59–60. doi: 10.1038/nmeth.3176 (cit. on p. 35).

205. B. Buchfink, K. Reuter, and H.-G. Drost: Sensitive Protein Alignments at Tree-of-Life Scale Using DIAMOND. *Nature Methods* **18**(4) (4 Apr. 2021), 366–368. doi: 10.1038/s41592-021-01101-x (cit. on p. 35).

206. W. R. Pearson and D. J. Lipman: Improved Tools for Biological Sequence Comparison. *Proceedings of the National Academy of Sciences of the United States of America* **85**(8) (Apr. 1988), 2444–2448. doi: 10.1073/pnas.85.8.2444 (cit. on p. 36).

207. D. J. Lipman and W. R. Pearson: Rapid and Sensitive Protein Similarity Searches. *Science* **227**(4693) (Mar. 22, 1985), 1435–1441. doi: 10.1126/science.2983426 (cit. on p. 36).

208. G. V. Saripella, E. L. L. Sonnhammer, and K. Forslund: Benchmarking the next Generation of Homology Inference Tools. *Bioinformatics* **32**(17) (Sept. 9, 2016), 2636. doi: 10.1093/bioinformatics/btw305 (cit. on p. 36).

209. R. D. Finn et al.: The Pfam Protein Families Database: Towards a More Sustainable Future. *Nucleic Acids Research* **44** (Database issue Jan. 1, 2016), D279. doi: 10.1093/nar/gkv1344 (cit. on p. 36).

210. N. Essoussi and S. Fayech: A Comparison of Four Pair-Wise Sequence Alignment Methods. *Bioinformation* **2**(4) (Dec. 28, 2007), 166–168. doi: 10.6026/97320630002166 (cit. on p. 36).

211. E. G. Shpaer et al.: Sensitivity and Selectivity in Protein Similarity Searches: A Comparison of Smith–Waterman in Hardware to BLAST and FASTA. *Genomics* **38**(2) (Dec. 1, 1996), 179–191. doi: 10.1006/geno.1996.0614 (cit. on p. 36).

212. S. Schleimer, D. S. Wilkerson, and A. Aiken: Winnowing: Local Algorithms for Document Fingerprinting. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA, June 9, 2003, 76–85. doi: 10.1145/872757.872770 (cit. on p. 36).

213. M. Roberts, W. Hayes, B. R. Hunt, S. M. Mount, and J. A. Yorke: Reducing Storage Requirements for Biological Sequence Comparison. *Bioinformatics* **20**(18) (Dec. 12, 2004), 3363–3369. doi: 10.1093/bioinformatics/bth408 (cit. on pp. 36, 37).

214. H. Li: Minimap and Miniasm: Fast Mapping and de Novo Assembly for Noisy Long Sequences. *Bioinformatics* **32**(14) (July 15, 2016), 2103–2110. doi: 10.1093/bioinformatics/btw152 (cit. on p. 36).

215. C. Jain, S. Koren, A. Dilthey, A. M. Phillippy, and S. Aluru: A Fast Adaptive Algorithm for Computing Whole-Genome Homology Maps. *Bioinformatics* **34**(17) (Sept. 1, 2018), i748–i756. doi: 10.1093/bioinformatics/bty597 (cit. on p. 36).

216. Y. Orenstein, D. Pellow, G. Marçais, R. Shamir, and C. Kingsford: Compact Universal K-Mer Hitting Sets. *Algorithms in Bioinformatics*. Cham, 2016, 257–268. doi: 10.1007/978-3-319-43681-4_21 (cit. on p. 37).

217. G. Marçais et al.: Improving the Performance of Minimizers and Winnowing Schemes. *Bioinformatics* **33**(14) (July 15, 2017), i110–i117. doi: 10.1093/bioinformatics/btx235 (cit. on p. 37).

218. R. Chikhi, A. Limasset, S. Jackman, J. T. Simpson, and P. Medvedev: On the Representation of de Bruijn Graphs. *Research in Computational Molecular Biology*. Cham, 2014, 35–55. doi: 10.1007/978-3-319-05269-4_4 (cit. on p. 37).

219. R. Edgar: Syncmers Are More Sensitive than Minimizers for Selecting Conserved K-mers in Biological Sequences. *PeerJ* **9** (Feb. 5, 2021), e10805. doi: 10.7717/peerj.10805 (cit. on p. 37).

220. K. Sahlin: Effective Sequence Similarity Detection with Strobemers. *Genome Research* **31**(11) (Jan. 11, 2021), 2080–2094. doi: 10.1101/gr.275648.121 (cit. on p. 37).

221. K. Sahlin: *Flexible Seed Size Enables Ultra-Fast and Accurate Read Alignment*. May 25, 2022. doi: 10.1101/2021.06.18.449070 (cit. on p. 37).

222. P. Weiner: Linear Pattern Matching Algorithms. *14th Annual Symposium on Switching and Automata Theory (Swat 1973)*. Oct. 1973, 1–11. doi: 10.1109/swat.1973.13 (cit. on p. 37).

223. U. Manber and G. Myers: Suffix Arrays: A New Method for On-Line String Searches. *SIAM Journal on Computing* **22**(5) (Oct. 1993), 935–948. doi: 10.1137/0222058 (cit. on pp. 37, 38).

224. M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch: The Enhanced Suffix Array and Its Applications to Genome Analysis. *Algorithms in Bioinformatics*. Berlin, Heidelberg, 2002, 449–463. doi: 10.1007/3-540-45784-4_35 (cit. on p. 37).

225. P. Ferragina and G. Manzini: Opportunistic Data Structures with Applications. *Proceedings 41st Annual Symposium on Foundations of Computer Science*. Nov. 2000, 390–398. doi: 10.1109/sfcs.2000.892127 (cit. on pp. 37, 38).

226. N. Bray, I. Dubchak, and L. Pachter: AVID: A Global Alignment Program. *Genome Research* **13**(1) (Jan. 1, 2003), 97–102. doi: 10.1101/gr.789803 (cit. on p. 37).

227. A. L. Delcher, A. Phillippy, J. Carlton, and S. L. Salzberg: Fast Algorithms for Large-Scale Genome Alignment and Comparison. *Nucleic Acids Research* **30**(11) (June 1, 2002), 2478–2483. doi: 10.1093/nar/30.11.2478 (cit. on p. 38).

228. M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch: Replacing Suffix Trees with Enhanced Suffix Arrays. *Journal of Discrete Algorithms* **2**(1) (Mar. 1, 2004), 53–86. doi: 10.1016/s1570-8667(03)00065-0 (cit. on p. 38).

229. G. Marçais et al.: MUMmer4: A Fast and Versatile Genome Alignment System. *PLOS Computational Biology* **14**(1) (Jan. 26, 2018), e1005944. doi: 10.1371/journal.pcbi.1005944 (cit. on p. 38).

230. E. M. McCreight: A Space-Economical Suffix Tree Construction Algorithm. *Journal of the ACM* **23**(2) (Apr. 1, 1976), 262–272. doi: 10.1145/321941.321946 (cit. on p. 38).

231. M. Burrows and D. Wheeler: *A Block-Sorting Lossless Data Compression Algorithm*. Digital Src Research Report, 1994. https://www.cs.jhu.edu/%7Elangmea/resources/burrows_wheeler.pdf (cit. on p. 38).

232. M. Vyverman, B. De Baets, V. Fack, and P. Dawyndt: Prospects and Limitations of Full-Text Index Structures in Genome Analysis. *Nucleic Acids Research* **40**(15) (Aug. 1, 2012), 6993–7015. doi: 10.1093/nar/gks408 (cit. on p. 38).

233. H. Cheng, M. Wu, and Y. Xu: FMtree: A Fast Locating Algorithm of FM-indexes for Genomic Data. *Bioinformatics* **34**(3) (Feb. 1, 2018), 416–424. doi: 10.1093/bioinformatics/btx596 (cit. on p. 38).

234. T. W. Lam, W. K. Sung, S. L. Tam, C. K. Wong, and S. M. Yiu: Compressed Indexing and Local Alignment of DNA. *Bioinformatics* **24**(6) (Mar. 15, 2008), 791–797. doi: 10.1093/bioinformatics/btn032 (cit. on pp. 38, 39).

235. H. Li and R. Durbin: Fast and Accurate Short Read Alignment with Burrows–Wheeler Transform. *Bioinformatics* **25**(14) (July 15, 2009), 1754–1760. doi: 10.1093/bioinformatics/btp324 (cit. on pp. 38–40).

236. H. Li and R. Durbin: Fast and Accurate Long-Read Alignment with Burrows–Wheeler Transform. *Bioinformatics* **26**(5) (Mar. 1, 2010), 589–595. doi: 10.1093/bioinformatics/btp698 (cit. on pp. 38, 39).

237. H. Li: Aligning Sequence Reads, Clone Sequences and Assembly Contigs with BWA-MEM. May 26, 2013. http://arxiv.org/abs/1303.3997 (cit. on pp. 38, 39).

238. Y. Liu and B. Schmidt: Long Read Alignment Based on Maximal Exact Match Seeds. *Bioinformatics* **28**(18) (Sept. 15, 2012), i318–i324. doi: 10.1093/bioinformatics/bts414 (cit. on pp. 38, 39).

239. B. Langmead and S. L. Salzberg: Fast Gapped-Read Alignment with Bowtie 2. *Nature Methods* **9**(4) (4 Apr. 2012), 357–359. doi: 10.1038/nmeth.1923 (cit. on pp. 38–40).

240. B. Song et al.: AnchorWave: Sensitive Alignment of Genomes with High Sequence Diversity, Extensive Structural Polymorphism, and Whole-Genome Duplication. *Proceedings of the National Academy of Sciences* **119**(1) (Jan. 4, 2022), e2113075119. doi: 10.1073/pnas.2113075119 (cit. on p. 38).

241. R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, 1998. doi: 10.1017/cbo9780511790492 (cit. on p. 38).

242. J. Söding: Protein Homology Detection by HMM-HMM Comparison. *Bioinformatics (Oxford, England)* **21**(7) (Apr. 1, 2005), 951–960. doi: 10.1093/bioinformatics/bti125 (cit. on p. 38).

243. R. D. Finn, J. Clements, and S. R. Eddy: HMMER Web Server: Interactive Sequence Similarity Searching. *Nucleic Acids Research* **39** (Web Server issue July 1, 2011), W29–w37. doi: 10.1093/nar/gkr367 (cit. on pp. 38, 43).

244. J. Wang, P. D. Keightley, and T. Johnson: MCALIGN2: Faster, Accurate Global Pairwise Alignment of Non-Coding DNA Sequences Based on Explicit Models of Indel Evolution. *BMC Bioinformatics* **7**(1) (June 8, 2006), 292. doi: 10.1186/1471-2105-7-292 (cit. on p. 38).

245. M. Ruffalo, T. LaFramboise, and M. Koyutürk: Comparative Analysis of Algorithms for Next-Generation Sequencing Read Alignment. *Bioinformatics (Oxford, England)* **27**(20) (Oct. 15, 2011), 2790–2796. doi: 10.1093/bioinformatics/btr477 (cit. on pp. 39, 40).

246. S. Schbath et al.: Mapping Reads on a Genomic Sequence: An Algorithmic Overview and a Practical Comparative Analysis. *Journal of Computational Biology* **19**(6) (June 2012), 796–813. doi: 10.1089/cmb.2012.0022 (cit. on p. 39).

247. A. Hatem, D. Bozdağ, A. E. Toland, and Ü. V. Çatalyürek: Benchmarking Short Sequence Mapping Tools. *BMC Bioinformatics* **14**(1) (June 7, 2013), 184. doi: 10.1186/1471-2105-14-184 (cit. on p. 39).

248. S. Canzar and S. L. Salzberg: Short Read Mapping: An Algorithmic Tour. *Proceedings of the IEEE* **105**(3) (Mar. 2017), 436–458. doi: 10.1109/jproc.2015.2455551 (cit. on p. 39).

249. M. Alser et al.: Technology Dictates Algorithms: Recent Developments in Read Alignment. *Genome Biology* **22**(1) (Aug. 26, 2021), 249. doi: 10.1186/s13059-021-02443-7 (cit. on pp. 39, 40).

250. K. Břinda, V. Boeva, and G. Kucherov: RNF: A General Framework to Evaluate NGS Read Mappers. *Bioinformatics* **32**(1) (Jan. 1, 2016), 136–139. doi: 10.1093/bioinformatics/btv524 (cit. on p. 39).

251. H.-N. Lin and W.-L. Hsu: Kart: A Divide-and-Conquer Algorithm for NGS Read Alignment. *Bioinformatics (Oxford, England)* **33**(15) (Aug. 1, 2017), 2281–2287. doi: 10.1093/bioinformatics/btx189 (cit. on p. 39).

252. C. B. Olson et al.: Hardware Acceleration of Short Read Mapping. *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*. Apr. 2012, 161–168. doi: 10.1109/fccm.2012.36 (cit. on p. 39).

253. P. Chen, C. Wang, X. Li, and X. Zhou: Accelerating the Next Generation Long Read Mapping with the FPGA-Based System. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **11**(5) (Sept. 2014), 840–852. doi: 10.1109/tcbb.2014.2326876 (cit. on p. 39).

254. H. Suzuki and M. Kasahara: Introducing Difference Recurrence Relations for Faster Semi-Global Alignment of Long Sequences. *BMC Bioinformatics* **19**(1) (Feb. 19, 2018), 45. doi: 10.1186/s12859-018-2014-8 (cit. on p. 39).

255. A. Zeni et al.: LOGAN: High-Performance GPU-Based X-Drop Long-Read Alignment. *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. May 2020, 462–471. doi: 10.1109/ipdps47924.2020.00055 (cit. on p. 39).

256. M. J. Chaisson and G. Tesler: Mapping Single Molecule Sequencing Reads Using Basic Local Alignment with Successive Refinement (BLASR): Application and Theory. *BMC Bioinformatics* **13**(1) (Sept. 19, 2012), 238. doi: 10.1186/1471-2105-13-238 (cit. on p. 39).

257. E. Haghshenas, S. C. Sahinalp, and F. Hach: lordFAST: Sensitive and Fast Alignment Search Tool for LOng Noisy Read Sequencing Data. *Bioinformatics (Oxford, England)* **35**(1) (Jan. 1, 2019), 20–27. doi: 10.1093/bioinformatics/bty544 (cit. on p. 39).

258. I. Sović et al.: Fast and Sensitive Mapping of Nanopore Sequencing Reads with GraphMap. *Nature Communications* **7**(1) (1 Apr. 15, 2016), 11307. doi: 10.1038/ncomms11307 (cit. on p. 39).

259. F. J. Sedlazeck et al.: Accurate Detection of Complex Structural Variations Using Single-Molecule Sequencing. *Nature Methods* **15**(6) (6 June 2018), 461–468. doi: 10.1038/s41592-018-0001-7 (cit. on p. 39).

260. C. Jain, A. Dilthey, S. Koren, S. Aluru, and A. M. Phillippy: A Fast Approximate Algorithm for Mapping Long Reads to Large Reference Databases. *Journal of Computational Biology* **25**(7) (July 1, 2018), 766–779. doi: 10.1089/cmb.2018.0036 (cit. on p. 39).

261. T. Prodanov and V. Bansal: Sensitive Alignment Using Paralogous Sequence Variants Improves Long-Read Mapping and Variant Calling in Segmental Duplications. *Nucleic Acids Research* **48**(19) (Nov. 4, 2020), e114. doi: 10.1093/nar/gkaa829 (cit. on pp. 39, 61).

262. C. Jain, A. Rhie, N. F. Hansen, S. Koren, and A. M. Phillippy: Long-Read Mapping to Repetitive Reference Sequences Using Winnowmap2. *Nature Methods* **19**(6) (6 June 2022), 705–710. doi: 10.1038/s41592-022-01457-8 (cit. on p. 40).

263. A. Mikheenko, A. V. Bzikadze, A. Gurevich, K. H. Miga, and P. A. Pevzner: TandemTools: Mapping Long Reads and Assessing/Improving Assembly Quality in Extra-Long Tandem Repeats. *Bioinformatics* **36** (Supplement_1 July 1, 2020), i75–i83. doi: 10.1093/bioinformatics/btaa440 (cit. on pp. 40, 56, 64).

264. H. Li, J. Ruan, and R. Durbin: Mapping Short DNA Sequencing Reads and Calling Variants Using Mapping Quality Scores. *Genome Research* **18**(11) (Jan. 11, 2008), 1851–1858. doi: 10.1101/gr.078212.108 (cit. on p. 40).

265. H. Li et al.: The Sequence Alignment/Map Format and SAMtools. *Bioinformatics (Oxford, England)* **25**(16) (Aug. 15, 2009), 3. doi: 10.1093/bioinformatics/btp352 (cit. on p. 40).

266. *Understanding MAPQ Scores in SAM Files: Does 37 = 42?* ACGT blog. http://www.acgt.me/blog/2014/12/16/understanding-mapq-scores-in-sam-files-does-37-42 (cit. on p. 40).

267. H. Lee and M. C. Schatz: Genomic Dark Matter: The Reliability of Short Read Mapping Illustrated by the Genome Mappability Score. *Bioinformatics* **28**(16) (Aug. 15, 2012), 2097–2105. doi: 10.1093/bioinformatics/bts330 (cit. on p. 40).

268. B. Langmead: A Tandem Simulation Framework for Predicting Mapping Quality. *Genome Biology* **18**(1) (Aug. 10, 2017), 152. doi: 10.1186/s13059-017-1290-3 (cit. on p. 40).

269. M. Ruffalo, M. Koyutürk, S. Ray, and T. LaFramboise: Accurate Estimation of Short Read Mapping Quality for Next-Generation Genome Sequencing. *Bioinformatics* **28**(18) (Sept. 15, 2012), i349–i355. doi: 10.1093/bioinformatics/bts408 (cit. on p. 40).

270. *Multiple Sequence Alignment Methods*. Vol. 1079. Totowa, NJ, 2014. doi: 10.1007/978-1-62703-646-7 (cit. on pp. 40, 42).

271. L. Wang and T. Jiang: On the Complexity of Multiple Sequence Alignment. *Journal of Computational Biology* **1**(4) (Jan. 1994), 337–348. doi: 10.1089/cmb.1994.1.337 (cit. on p. 41).

272. W. Just: Computational Complexity of Multiple Sequence Alignment with SP-Score. *Journal of Computational Biology* **8**(6) (Nov. 2001), 615–623. doi: 10.1089/106652701753307511 (cit. on p. 41).

273. F. Tang et al.: HAlign 3: Fast Multiple Alignment of Ultra-Large Numbers of Similar DNA/RNA Sequences. *Molecular Biology and Evolution* **39**(8) (Aug. 1, 2022), msac166. doi: 10.1093/molbev/msac166 (cit. on p. 41).

274. D.-F. Feng and R. F. Doolittle: Progressive Sequence Alignment as a Prerequisitetto Correct Phylogenetic Trees. *Journal of Molecular Evolution* **25**(4) (Aug. 1, 1987), 351–360. doi: 10.1007/bf02603120 (cit. on pp. 41, 42).

275. D. T. Jones, W. R. Taylor, and J. M. Thornton: The Rapid Generation of Mutation Data Matrices from Protein Sequences. *Bioinformatics* **8**(3) (June 1, 1992), 275–282. doi: 10.1093/bioinformatics/8.3.275 (cit. on p. 41).

276. B. E. Blaisdell: A Measure of the Similarity of Sets of Sequences Not Requiring Sequence Alignment. *Proceedings of the National Academy of Sciences* **83**(14) (July 1986), 5155–5159. doi: 10.1073/pnas.83.14.5155 (cit. on p. 41).

277. I. Gronau and S. Moran: Optimal Implementations of UPGMA and Other Common Clustering Algorithms. *Information Processing Letters* **104**(6) (Dec. 16, 2007), 205–210. doi: 10.1016/j.ipl.2007.07.002 (cit. on p. 41).

278. N. Saitou and M. Nei: The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees. *Molecular Biology and Evolution* **4**(4) (July 1, 1987), 406–425. doi: 10.1093/oxfordjournals.molbev.a040454 (cit. on p. 41).

279. K. Katoh and H. Toh: PartTree: An Algorithm to Build an Approximate Tree from a Large Number of Unaligned Sequences. *Bioinformatics* **23**(3) (Feb. 1, 2007), 372–374. doi: 10.1093/bioinformatics/btl592 (cit. on p. 42).

280. F. Sievers et al.: Fast, Scalable Generation of High-Quality Protein Multiple Sequence Alignments Using Clustal Omega. *Molecular Systems Biology* **7**(1) (Jan. 2011), 539. doi: 10.1038/msb.2011.75 (cit. on pp. 42, 43).

281. G. Blackshields, F. Sievers, W. Shi, A. Wilm, and D. G. Higgins: Sequence Embedding for Fast Construction of Guide Trees for Multiple Sequence Alignment. *Algorithms for Molecular Biology* **5**(1) (May 14, 2010), 21. doi: 10.1186/1748-7188-5-21 (cit. on p. 42).

282. S. F. Altschul: Gap Costs for Multiple Sequence Alignment. *Journal of Theoretical Biology* **138**(3) (June 3, 1989), 297–309. doi: 10.1016/s0022-5193(89)80196-1 (cit. on p. 42).

283. S. F. Altschul, R. J. Carroll, and D. J. Lipman: Weights for Data Related by a Tree. *Journal of Molecular Biology* **207**(4) (June 20, 1989), 647–653. doi: 10.1016/0022-2836(89)90234-9 (cit. on p. 42).

284. R. C. Edgar and K. Sjölander: A Comparison of Scoring Functions for Protein Sequence Profile Alignment. *Bioinformatics* **20**(8) (May 22, 2004), 1301–1308. doi: 10.1093/bioinformatics/bth090 (cit. on p. 42).

285. C. Notredame, L. Holm, and D. G. Higgins: COFFEE: An Objective Function for Multiple Sequence Alignments. *Bioinformatics* **14**(5) (June 1, 1998), 407–422. doi: 10.1093/bioinformatics/14.5.407 (cit. on p. 42).

286. C. Notredame, D. G. Higgins, and J. Heringa: T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence alignment11Edited by J. Thornton. *Journal of Molecular Biology* **302**(1) (Sept. 8, 2000), 205–217. doi: 10.1006/jmbi.2000.4042 (cit. on p. 43).

287. R. C. Edgar: MUSCLE: A Multiple Sequence Alignment Method with Reduced Time and Space Complexity. *BMC Bioinformatics* **5**(1) (Aug. 19, 2004), 113. doi: 10.1186/1471-2105-5-113 (cit. on p. 43).

288. R. C. Edgar: MUSCLE: Multiple Sequence Alignment with High Accuracy and High Throughput. *Nucleic Acids Research* **32**(5) (Mar. 1, 2004), 1792–1797. doi: 10.1093/nar/gkh340 (cit. on p. 43).

289. C. B. Do, M. S. Mahabhashyam, M. Brudno, and S. Batzoglou: ProbCons: Probabilistic Consistency-Based Multiple Sequence Alignment. *Genome Research* **15**(2) (Feb. 2005), 330–340. doi: 10.1101/gr.2821705 (cit. on p. 43).

290. J. D. Thompson, D. G. Higgins, and T. J. Gibson: CLUSTAL W: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice. *Nucleic Acids Research* **22**(22) (Nov. 11, 1994), 4673–4680. doi: 10.1093/nar/22.22.4673 (cit. on p. 43).

291. J. D. Thompson, T. J. Gibson, F. Plewniak, F. Jeanmougin, and D. G. Higgins: The CLUSTAL_X Windows Interface: Flexible Strategies for Multiple Sequence Alignment Aided by Quality Analysis Tools. *Nucleic Acids Research* **25**(24) (Dec. 1, 1997), 4876–4882. doi: 10.1093/nar/25.24.4876 (cit. on p. 43).

292. Y. Liu, B. Schmidt, and D. L. Maskell: MSAProbs: Multiple Sequence Alignment Based on Pair Hidden Markov Models and Partition Function Posterior Probabilities. *Bioinformatics* **26**(16) (Aug. 15, 2010), 1958–1964. doi: 10.1093/bioinformatics/btq338 (cit. on p. 43).

293. F. Lemoine, L. Blassel, J. Voznica, and O. Gascuel: COVID-Align: Accurate Online Alignment of hCoV-19 Genomes Using a Profile HMM. *Bioinformatics* (btaa871 Oct. 12, 2020). doi: 10.1093/bioinformatics/btaa871 (cit. on p. 43).

294. S. R. Eddy: Multiple Alignment Using Hidden Markov Models. *International Conference on Intelligent Systems for Molecular Biology*. 1995, 7. https://www.aaai.org/Papers/ISMB/1995/ISMB95-014.pdf (cit. on p. 43).

295. J. Kim, S. Pramanik, and M. J. Chung: Multiple Sequence Alignment Using Simulated Annealing. *Bioinformatics* **10**(4) (July 1, 1994), 419–426. doi: 10.1093/bioinformatics/10.4.419 (cit. on p. 43).

296. M. Ishikawa et al.: Multiple Sequence Alignment by Parallel Simulated Annealing. *Bioinformatics* **9**(3) (June 1, 1993), 267–273. doi: 10.1093/bioinformatics/9.3.267 (cit. on p. 43).

297. H. Huo and V. Stojkovic: A Simulated Annealing Algorithm for Multiple Sequence Alignment with Guaranteed Accuracy. *Third International Conference on Natural Computation (ICNC 2007)*. **2**. Aug. 2007, 270–274. doi: 10.1109/icnc.2007.139 (cit. on p. 43).

298. B. Chowdhury and G. Garai: A Review on Multiple Sequence Alignment from the Perspective of Genetic Algorithm. *Genomics* **109**(5) (Oct. 1, 2017), 419–431. doi: 10.1016/j.ygeno.2017.06.007 (cit. on p. 43).

299. C. Zhang and A. K. Wong: A Genetic Algorithm for Multiple Molecular Sequence Alignment. *Bioinformatics* **13**(6) (Dec. 1, 1997), 565–581. doi: 10.1093/bioinformatics/13.6.565 (cit. on p. 43).

300. F. Naznin, R. Sarker, and D. Essam: Vertical Decomposition with Genetic Algorithm for Multiple Sequence Alignment. *BMC Bioinformatics* **12**(1) (Aug. 25, 2011), 353. doi: 10.1186/1471-2105-12-353 (cit. on p. 43).

301. F. Naznin, R. Sarker, and D. Essam: Progressive Alignment Method Using Genetic Algorithm for Multiple Sequence Alignment. *IEEE Transactions on Evolutionary Computation* **16**(5) (Oct. 2012), 615–631. doi: 10.1109/tevc.2011.2162849 (cit. on p. 43).

302. C. Notredame and D. G. Higgins: SAGA: Sequence Alignment by Genetic Algorithm. *Nucleic Acids Research* **24**(8) (Apr. 15, 1996), 1515–1524. doi: 10.1093/nar/24.8.1515 (cit. on p. 43).

303. I. Aksamentov, C. Roemer, E. Hodcroft, and R. Neher: Nextclade: Clade Assignment, Mutation Calling and Quality Control for Viral Genomes. *Journal of Open Source Software* **6**(67) (Nov. 30, 2021), 3773. doi: 10.21105/joss.03773 (cit. on p. 43).

304. E. Garriga et al.: Large Multiple Sequence Alignments with a Root-to-Leaf Regressive Method. *Nature Biotechnology* **37**(12) (12 Dec. 2019), 1466–1470. doi: 10.1038/s41587-019-0333-6 (cit. on p. 43).

305. C. Notredame: Recent Evolutions of Multiple Sequence Alignment Algorithms. *PLOS Computational Biology* **3**(8) (Aug. 31, 2007), e123. doi: 10.1371/journal.pcbi.0030123 (cit. on p. 44).

306. C. Notredame: Recent Progress in Multiple Sequence Alignment: A Survey. *Pharmacogenomics* **3**(1) (Jan. 2002), 131–144. doi: 10.1517/14622416.3.1.131 (cit. on p. 44).

307. R. C. Edgar and S. Batzoglou: Multiple Sequence Alignment. *Current Opinion in Structural Biology* **16**(3) (June 1, 2006), 368–373. doi: 10.1016/j.sbi.2006.04.004 (cit. on p. 44).

308. F. S.-M. Pais, P. d. C. Ruy, G. Oliveira, and R. S. Coimbra: Assessing the Efficiency of Multiple Sequence Alignment Programs. *Algorithms for Molecular Biology* **9**(1) (Mar. 6, 2014), 4. doi: 10.1186/1748-7188-9-4 (cit. on p. 44).

309. J. D. Thompson, F. Plewniak, and O. Poch: BAliBASE: A Benchmark Alignment Database for the Evaluation of Multiple Alignment Programs. *Bioinformatics* **15**(1) (Jan. 1, 1999), 87–88. doi: 10.1093/bioinformatics/15.1.87 (cit. on p. 44).

310. L. Bragg, G. Stone, M. Imelfort, P. Hugenholtz, and G. W. Tyson: Fast, Accurate Error-Correction of Amplicon Pyrosequences Using Acacia. *Nature Methods* **9**(5) (5 May 2012), 425–426. doi: 10.1038/nmeth.1990 (cit. on p. 47).

311. K. Sahlin and P. Medvedev: Error Correction Enables Use of Oxford Nanopore Technology for Reference-Free Transcriptome Analysis. *Nature Communications* **12**(1) (1 Jan. 4, 2021), 2. doi: 10.1038/s41467-020-20340-8 (cit. on p. 47).

312. H. Liu et al.: SMARTdenovo: A de Novo Assembler Using Long Noisy Reads. *Gigabyte* **2021** (Mar. 8, 2021), 1–9. doi: 10.46471/gigabyte.15 (cit. on p. 47).

313. R. L. Graham, D. E. Knuth, and O. Patashnik: *Concrete Mathematics: A Foundation for Computer Science*. 2nd ed. Reading, Mass, 1994. isbn: 978-0-201-55802-9 (cit. on p. 54).

314. M. D. Adams et al.: The Genome Sequence of Drosophila Melanogaster. *Science* **287**(5461) (Mar. 24, 2000), 2185–2195. doi: 10.1126/science.287.5461.2185 (cit. on pp. 56, 64).

315. A. Rhie, B. P. Walenz, S. Koren, and A. M. Phillippy: Merqury: Reference-Free Quality, Completeness, and Phasing Assessment for Genome Assemblies. *Genome Biology* **21**(1) (Sept. 14, 2020), 245. doi: 10.1186/s13059-020-02134-9 (cit. on p. 58).

316. H. Li: New Strategies to Improve Minimap2 Alignment Accuracy. *Bioinformatics* **37**(arXiv:2108.03515) (23 Dec. 1, 2021), 4572–4574. doi: 10.1093/bioinformatics/btab705 (cit. on p. 61).

317. H. Li et al.: A Synthetic-Diploid Benchmark for Accurate Variant-Calling Evaluation. *Nature Methods* **15**(8) (8 Aug. 2018), 595–597. doi: 10.1038/s41592-018-0054-7 (cit. on p. 61).

318. C. Yang, J. Chu, R. L. Warren, and I. Birol: NanoSim: Nanopore Sequence Read Simulator Based on Statistical Characterization. *GigaScience* **6**(4) (Apr. 1, 2017). doi: 10.1093/gigascience/gix010 (cit. on pp. 64, 143).

319. J. A. Martin and Z. Wang: Next-Generation Transcriptome Assembly. *Nature Reviews Genetics* **12**(10) (10 Oct. 2011), 671–682. doi: 10.1038/nrg3068 (cit. on p. 67).

320. M. Kyriakidou, H. H. Tai, N. L. Anglin, D. Ellis, and M. V. Strömvik: Current Strategies of Polyploid Plant Genome Sequence Assembly. *Frontiers in Plant Science* **9** (2018). doi: 10.3389/fpls.2018.01660 (cit. on p. 67).

321. K. Paszkiewicz and D. J. Studholme: De Novo Assembly of Short Sequence Reads. *Briefings in Bioinformatics* **11**(5) (Sept. 1, 2010), 457–472. doi: 10.1093/bib/bbq020 (cit. on p. 67).

322. J.-i. Sohn and J.-W. Nam: The Present and Future of de Novo Whole-Genome Assembly. *Briefings in Bioinformatics* **19**(1) (Jan. 1, 2018), 23–40. doi: 10.1093/bib/bbw096 (cit. on p. 67).

323. R. D. Sleator and P. Walsh: An Overview of in Silico Protein Function Prediction. *Archives of Microbiology* **192**(3) (Mar. 1, 2010), 151–155. doi: 10.1007/s00203-010-0549-9 (cit. on p. 67).

324. D. C. Koboldt: Best Practices for Variant Calling in Clinical Sequencing. *Genome Medicine* **12**(1) (Oct. 26, 2020), 91. doi: 10.1186/s13073-020-00791-w (cit. on p. 67).

325. C. Alkan, B. P. Coe, and E. E. Eichler: Genome Structural Variation Discovery and Genotyping. *Nature Reviews Genetics* **12**(5) (5 May 2011), 363–376. doi: 10.1038/nrg2958 (cit. on p. 67).

326. S. S. Ho, A. E. Urban, and R. E. Mills: Structural Variation in the Sequencing Era. *Nature Reviews Genetics* **21**(3) (3 Mar. 2020), 171–189. doi: 10.1038/s41576-019-0180-9 (cit. on p. 67).

327. D. A. Morrison: Phylogenetic Tree-Building. *International Journal for Parasitology* **26**(6) (June 1, 1996), 589–617. doi: 10.1016/0020-7519(96)00044-6 (cit. on p. 67).

328. P. Kapli, Z. Yang, and M. J. Telford: Phylogenetic Tree Building in the Genomic Age. *Nature Reviews Genetics* **21**(7) (7 July 2020), 428–444. doi: 10.1038/s41576-020-0233-0 (cit. on p. 67).

329. B. Kuhlman and P. Bradley: Advances in Protein Structure Prediction and Design. *Nature Reviews Molecular Cell Biology* **20**(11) (11 Nov. 2019), 681–697. doi: 10.1038/s41580-019-0163-x (cit. on p. 67).

330. M. Ammad-ud-din, S. A. Khan, K. Wennerberg, and T. Aittokallio: Systematic Identification of Feature Combinations for Predicting Drug Response with Bayesian Multi-View Multi-Task Linear Regression. *Bioinformatics* **33**(14) (July 15, 2017), i359–i368. doi: 10.1093/bioinformatics/btx266 (cit. on p. 68).

331. M. C. Steiner, K. M. Gibson, and K. A. Crandall: Drug Resistance Prediction Using Deep Learning Techniques on HIV-1 Sequence Data. en. *Viruses* **12**(5) (5 May 2020), 560. doi: 10.3390/v12050560 (cit. on pp. 68, 78, 101, 107, 128).

332. F. Noé, G. De Fabritiis, and C. Clementi: Machine Learning for Protein Folding and Dynamics. *Current Opinion in Structural Biology* **60** (Feb. 1, 2020), 77–84. doi: 10.1016/j.sbi.2019.12.005 (cit. on p. 68).

333. R. Pearce and Y. Zhang: Toward the Solution of the Protein Structure Prediction Problem. *Journal of Biological Chemistry* **297**(1) (July 1, 2021). doi: 10.1016/j.jbc.2021.100870 (cit. on p. 68).

334. K. Tunyasuvunakool et al.: Highly Accurate Protein Structure Prediction for the Human Proteome. *Nature* **596**(7873) (7873 Aug. 2021), 590–596. doi: 10.1038/s41586-021-03828-1 (cit. on p. 68).

335. J. Cheng, A. N. Tegge, and P. Baldi: Machine Learning Methods for Protein Structure Prediction. *IEEE Reviews in Biomedical Engineering* **1** (2008), 41–49. doi: 10.1109/rbme.2008.2008239 (cit. on p. 68).

336. M. AlQuraishi: Machine Learning in Protein Structure Prediction. *Current Opinion in Chemical Biology* **65** (Dec. 1, 2021), 1–8. doi: 10.1016/j.cbpa.2021.04.005 (cit. on p. 68).

337. B. J. Wittmann, K. E. Johnston, Z. Wu, and F. H. Arnold: Advances in Machine Learning for Directed Evolution. *Current Opinion in Structural Biology* **69** (Aug. 1, 2021), 11–18. doi: 10.1016/j.sbi.2021.01.008 (cit. on p. 68).

338. K. K. Yang, Z. Wu, and F. H. Arnold: Machine-Learning-Guided Directed Evolution for Protein Engineering. *Nature Methods* **16**(8) (8 Aug. 2019), 687–694. doi: 10.1038/s41592-019-0496-6 (cit. on p. 68).

339. G. Li, Y. Dong, and M. T. Reetz: Can Machine Learning Revolutionize Directed Evolution of Selective Enzymes? *Advanced Synthesis & Catalysis* **361**(11) (2019), 2377–2386. doi: 10.1002/adsc.201900149 (cit. on p. 68).

340. R. Xie, J. Wen, A. Quitadamo, J. Cheng, and X. Shi: A Deep Auto-Encoder Model for Gene Expression Prediction. *BMC Genomics* **18**(9) (Nov. 17, 2017), 845. doi: 10.1186/s12864-017-4226-0 (cit. on p. 68).

341. F. M. Ortuño et al.: Comparing Different Machine Learning and Mathematical Regression Models to Evaluate Multiple Sequence Alignments. *Neurocomputing* **164** (Sept. 21, 2015), 123–136. doi: 10.1016/j.neucom.2015.01.080 (cit. on p. 68).

342. S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu: Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model. *PLOS Computational Biology* **13**(1) (Jan. 5, 2017), e1005324. doi: 10.1371/journal.pcbi.1005324 (cit. on p. 68).

343. H. Haga et al.: A Machine Learning-Based Treatment Prediction Model Using Whole Genome Variants of Hepatitis C Virus. *Plos One* **15**(11) (Nov. 5, 2020), e0242028. doi: 10.1371/journal.pone.0242028 (cit. on p. 68).

344. M. Zazzi et al.: Predicting Response to Antiretroviral Treatment by Machine Learning: The EuResist Project. *Intervirology* **55**(2) (2012), 123–127. doi: 10.1159/000332008 (cit. on p. 68).

345. Y. Ren et al.: Prediction of Antimicrobial Resistance Based on Whole-Genome Sequencing and Machine Learning. *Bioinformatics* **38**(2) (Jan. 15, 2022), 325–334. doi: 10.1093/bioinformatics/btab681 (cit. on pp. 68, 80).

346. J. I. Kim et al.: Machine Learning for Antimicrobial Resistance Prediction: Current Practice, Limitations, and Clinical Perspective. *Clinical Microbiology Reviews* **0**(0) (May 25, 2022), e00179–21. doi: 10.1128/cmr.00179-21 (cit. on p. 68).

347. Y. Wang et al.: Predicting DNA Methylation State of CpG Dinucleotide Using Genome Topological Features and Deep Networks. *Scientific Reports* **6**(1) (1 Jan. 22, 2016), 19598. doi: 10.1038/srep19598 (cit. on p. 68).

348. G. Rätsch, S. Sonnenburg, and C. Schäfer: Learning Interpretable SVMs for Biological Sequence Classification. *BMC Bioinformatics* **7**(1) (Mar. 20, 2006), S9. doi: 10.1186/1471-2105-7-s1-s9 (cit. on p. 68).

349. D. T. Jones: Protein Secondary Structure Prediction Based on Position-Specific Scoring Matrices. *Journal of Molecular Biology* **292**(2) (Sept. 17, 1999), 195–202. doi: 10.1006/jmbi.1999.3091 (cit. on p. 69).

350. T. Alioto: Gene Prediction. Totowa, NJ, 2012, pp. 175–201. doi: 10.1007/978-1-61779-582-4_6 (cit. on p. 69).

351. Z. Fang et al.: PlasGUN: gene prediction in plasmid metagenomic short reads using deep learning. *Bioinformatics* **36**(10) (May 1, 2020), 3239–3241. doi: 10.1093/bioinformatics/btaa103 (cit. on p. 69).

352. L. Wei, Y. Ding, R. Su, J. Tang, and Q. Zou: Prediction of Human Protein Subcellular Localization Using Deep Learning. *Journal of Parallel and Distributed Computing* **117** (July 1, 2018), 212–217. doi: 10.1016/j.jpdc.2017.08.009 (cit. on pp. 69, 128).

353. H. Wang, L. Yan, H. Huang, and C. Ding: From Protein Sequence to Protein Function via Multi-Label Linear Discriminant Analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **14**(3) (May 2017), 503–513. doi: 10.1109/tcbb.2016.2591529 (cit. on p. 69).

354. D. R. Kelley, J. Snoek, and J. L. Rinn: Basset: Learning the Regulatory Code of the Accessible Genome with Deep Convolutional Neural Networks. *Genome Research* **26**(7) (July 2016), 990–999. doi: 10.1101/gr.200535.115 (cit. on p. 69).

355. T. Hastie, R. Tibshirani, and J. Friedman: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. en. Aug. 26, 2009. isbn: 978-0-387-84858-7 (cit. on pp. 69, 71, 72, 74, 76, 111).

356. E. V. Kriventseva, M. Biswas, and R. Apweiler: Clustering and Analysis of Protein Families. *Current Opinion in Structural Biology* **11**(3) (June 1, 2001), 334–339. doi: 10.1016/s0959-440x(00)00211-6 (cit. on p. 69).

357. L. Fu, B. Niu, Z. Zhu, S. Wu, and W. Li: CD-HIT: Accelerated for Clustering the next-Generation Sequencing Data. *Bioinformatics* **28**(23) (Dec. 1, 2012), 3150–3152. doi: 10.1093/bioinformatics/bts565 (cit. on p. 69).

358. M. Balaban, N. Moshiri, U. Mai, X. Jia, and S. Mirarab: TreeCluster: Clustering Biological Sequences Using Phylogenetic Trees. *Plos One* **14**(8) (Aug. 22, 2019), e0221068. doi: 10.1371/journal.pone.0221068 (cit. on p. 69).

359. E. Zorita, P. Cuscó, and G. J. Filion: Starcode: Sequence Clustering Based on All-Pairs Search. *Bioinformatics* **31**(12) (June 15, 2015), 1913–1919. doi: 10.1093/bioinformatics/btv053 (cit. on p. 69).

360. B. D. Ondov et al.: Mash: Fast Genome and Metagenome Distance Estimation Using MinHash. *Genome Biology* **17**(1) (June 20, 2016), 132. doi: 10.1186/s13059-016-0997-x (cit. on p. 70).

361. D. N. Baker and B. Langmead: Dashing: Fast and Accurate Genomic Distances with HyperLogLog. *Genome Biology* **20**(1) (Dec. 4, 2019), 265. doi: 10.1186/s13059-019-1875-0 (cit. on p. 70).

362. G. Corso et al.: Neural Distance Embeddings for Biological Sequences. *Advances in Neural Information Processing Systems*. **34**. 2021, 18539–18551. https://proceedings.neurips.cc/paper/2021/hash/9a1de01f893e0d2551ecbb7ce4dc963e-Abstract.html (cit. on p. 70).

363. T. A. Hopf et al.: Mutation Effects Predicted from Sequence Co-Variation. *Nature Biotechnology* **35**(2) (2 Feb. 2017), 128–135. doi: 10.1038/nbt.3769 (cit. on p. 70).

364. B. M. Castro, R. B. Lemes, J. Cesar, T. Hünemeier, and F. Leonardi: A Model Selection Approach for Multiple Sequence Segmentation and Dimensionality Reduction. *Journal of Multivariate Analysis* **167** (Sept. 1, 2018), 319–330. doi: 10.1016/j.jmva.2018.05.006 (cit. on p. 70).

365. T. Haschka, L. Ponger, C. Escudé, and J. Mozziconacci: MNHN-Tree-Tools: A Toolbox for Tree Inference Using Multi-Scale Clustering of a Set of Sequences. *Bioinformatics* **37**(21) (Nov. 1, 2021), 3947–3949. doi: 10.1093/bioinformatics/btab430 (cit. on p. 70).

366. T. Konishi et al.: Principal Component Analysis Applied Directly to Sequence Matrix. *Scientific Reports* **9**(1) (1 Dec. 17, 2019), 19297. doi: 10.1038/s41598-019-55253-0 (cit. on p. 70).

367. A. Ben-Hur and I. Guyon: Detecting Stable Clusters Using Principal Component Analysis. *Functional Genomics: Methods and Protocols*. Totowa, NJ, 2003, 159–182. doi: 10.1385/1-59259-364-x:159 (cit. on p. 70).

368. C. Ding and X. He: K-Means Clustering via Principal Component Analysis. *Proceedings of the Twenty-First International Conference on Machine Learning*. New York, NY, USA, July 4, 2004, 29. doi: 10.1145/1015330.1015408 (cit. on p. 70).

369. G. Casari, C. Sander, and A. Valencia: Sequencespace: A Tool for Family Analysis. *Nature Structural Biology* **2**(2) (Feb. 1995), 171–178. doi: 10.1038/nsb0295-171 (cit. on p. 70).

370. M. Clamp, J. Cuff, S. M. Searle, and G. J. Barton: The Jalview Java Alignment Editor. *Bioinformatics* **20**(3) (Feb. 12, 2004), 426–427. doi: 10.1093/bioinformatics/btg430 (cit. on p. 70).

371. Z. Xia, L.-Y. Wu, X. Zhou, and S. T. Wong: Semi-Supervised Drug-Protein Interaction Prediction from Heterogeneous Biological Spaces. *BMC Systems Biology* **4**(2) (Sept. 13, 2010), S6. doi: 10.1186/1752-0509-4-s2-s6 (cit. on p. 70).

372. I. A. Tamposis, K. D. Tsirigos, M. C. Theodoropoulou, P. I. Kontou, and P. G. Bagos: Semi-Supervised Learning of Hidden Markov Models for Biological Sequence Analysis. *Bioinformatics* **35**(13) (July 1, 2019), 2208–2215. doi: 10.1093/bioinformatics/bty910 (cit. on p. 70).

373. A. Elnaggar et al.: *ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing*. May 4, 2021. doi: 10.48550/arXiv.2007.06225 (cit. on pp. 70, 132).

374. A. X. Lu et al.: Discovering Molecular Features of Intrinsically Disordered Regions by Using Evolution for Contrastive Learning. *PLOS Computational Biology* **18**(6) (June 29, 2022), e1010238. doi: 10.1371/journal.pcbi.1010238 (cit. on p. 70).

375. R. Townshend, R. Bedi, P. Suriana, and R. Dror: End-to-End Learning on 3D Protein Structure for Interface Prediction. *Advances in Neural Information Processing Systems*. **32**. 2019. https://proceedings.neurips.cc/paper/2019/hash/6c7de1f27f7de61a6daddfffbe05c058-Abstract.html (cit. on p. 70).

376. B. Lee, J. Baek, S. Park, and S. Yoon: deepTarget: End-to-end Learning Framework for microRNA Target Prediction Using Deep Recurrent Neural Networks. *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. New York, NY, USA, Oct. 2, 2016, 434–442. doi: 10.1145/2975167.2975212 (cit. on p. 70).

377. I. Goodfellow, Y. Bengio, and A. Courville: *Deep Learning*. 2016. http://www.deeplearningbook.org (cit. on p. 71).

378. Q. Wang, Y. Ma, K. Zhao, and Y. Tian: A Comprehensive Survey of Loss Functions in Machine Learning. *Annals of Data Science* **9**(2) (Apr. 1, 2022), 187–212. doi: 10.1007/s40745-020-00253-5 (cit. on p. 71).

379. Y. Jiao and P. Du: Performance Measures in Evaluating Machine Learning Based Bioinformatics Predictors for Classifications. *Quantitative Biology* **4**(4) (Dec. 1, 2016), 320–330. doi: 10.1007/s40484-016-0081-2 (cit. on p. 71).

380. K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann: The Balanced Accuracy and Its Posterior Distribution. *2010 20th International Conference on Pattern Recognition*. Aug. 2010, 3121–3124. doi: 10.1109/icpr.2010.764 (cit. on pp. 72, 112, 158).

381. S. Kaufman, S. Rosset, and C. Perlich: Leakage in Data Mining: Formulation, Detection, and Avoidance. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA, Aug. 21, 2011, 556–563. doi: 10.1145/2020408.2020496 (cit. on p. 72).

382. S. Whalen, J. Schreiber, W. S. Noble, and K. S. Pollard: Navigating the Pitfalls of Applying Machine Learning in Genomics. *Nature Reviews Genetics* **23**(3) (3 Mar. 2022), 169–181. doi: 10.1038/s41576-021-00434-9 (cit. on p. 72).

383. R. A. Fisher: On the Interpretation of X2 from Contingency Tables, and the Calculation of P. *Journal of the Royal Statistical Society* **85**(1) (1922), 87–94. doi: 10.2307/2340521 (cit. on p. 73).

384. K. Pearson: X. On the Criterion That a given System of Deviations from the Probable in the Case of a Correlated System of Variables Is Such That It Can Be Reasonably Supposed to Have Arisen from Random Sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **50**(302) (July 1, 1900), 157–175. doi: 10.1080/14786440009463897 (cit. on p. 73).

385. A. E. Hoerl and R. W. Kennard: Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* **12**(1) (Feb. 1, 1970), 55–67. doi: 10.1080/00401706.1970.10488634 (cit. on p. 75).

386. R. Tibshirani: Regression Shrinkage and Selection Via the Lasso. en. *Journal of the Royal Statistical Society: Series B (Methodological)* **58**(1) (1996), 267–288. doi: 10.1111/j.2517-6161.1996.tb02080.x (cit. on pp. 75, 110, 111).

387. H. Zhang: The Optimality of Naive Bayes. *Proceedings of the the 17th International FLAIRS conference (FLAIRS2004)*. 2004, 6. https://www.aaai.org/Papers/FLAIRS/2004/Flairs04-097.pdf (cit. on p. 75).

388. I. Rish: An Empirical Study of the Naive Bayes Classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence*. **3**. 22. 2001, 6. https://www.cc.gatech.edu/home/isbell/classes/reading/papers/Rish.pdf (cit. on p. 75).

389. V. Vapnik: *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Berlin, Heidelberg, 1982. isbn: 978-0-387-90733-8 (cit. on p. 75).

390. B. E. Boser, I. M. Guyon, and V. N. Vapnik: A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. New York, NY, USA, July 1, 1992, 144–152. doi: 10.1145/130385.130401 (cit. on p. 75).

391. C. Cortes and V. Vapnik: Support-Vector Networks. *Machine Learning* **20**(3) (Sept. 1, 1995), 273–297. doi: 10.1007/bf00994018 (cit. on p. 75).

392. H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik: Support Vector Regression Machines. *Advances in Neural Information Processing Systems*. **9**. 1996. https://proceedings.neurips.cc/paper/1996/hash/d38901788c533e8286cb6400b40b386d-Abstract.html (cit. on p. 75).

393. L. Breiman: Random Forests. en. *Machine Learning* **45**(1) (Oct. 1, 2001), 5–32. doi: 10.1023/a:1010933404324 (cit. on pp. 75, 111).

394. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone: *Classification and Regression Trees*. 1983. doi: 10.1201/9781315139470 (cit. on p. 75).

395. C. Kingsford and S. L. Salzberg: What Are Decision Trees? *Nature Biotechnology* **26**(9) (9 Sept. 2008), 1011–1013. doi: 10.1038/nbt0908-1011 (cit. on p. 76).

396. R. Caruana and A. Niculescu-Mizil: An Empirical Comparison of Supervised Learning Algorithms. *Proceedings of the 23rd International Conference on Machine Learning*. New York, NY, USA, June 25, 2006, 161–168. doi: 10.1145/1143844.1143865 (cit. on p. 77).

397. P. Yang, Y. Hwa Yang, B. B. Zhou, and A. Y. Zomaya: A Review of Ensemble Methods in Bioinformatics. *Current Bioinformatics* **5**(4) (Dec. 1, 2010), 296–308. doi: 10.2174/157489310794072508 (cit. on p. 77).

398. K. Potdar, T. S., and C. D.: A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. *International Journal of Computer Applications* **175**(4) (Oct. 17, 2017), 7–9. doi: 10.5120/ijca2017915495 (cit. on pp. 78, 79).

399. H. Hassani Saadi, R. Sameni, and A. Zollanvari: Interpretive Time-Frequency Analysis of Genomic Sequences. *BMC Bioinformatics* **18**(4) (Mar. 22, 2017), 154. doi: 10.1186/s12859-017-1524-0 (cit. on p. 78).

400. R. K. Brouwer: A Feed-Forward Network for Input That Is Both Categorical and Quantitative. *Neural Networks* **15**(7) (Sept. 1, 2002), 881–890. doi: 10.1016/s0893-6080(02)00090-4 (cit. on p. 78).

401. K. Kunanbayev, I. Temirbek, and A. Zollanvari: Complex Encoding. *2021 International Joint Conference on Neural Networks (IJCNN)*. July 2021, 1–6. doi: 10.1109/ijcnn52387.2021.9534094 (cit. on p. 78).

402. Y. Dufresne et al.: The K-mer File Format: A Standardized and Compact Disk Representation of Sets of k-Mers. *Bioinformatics* (July 29, 2022), btac528. doi: 10.1093/bioinformatics/btac528 (cit. on p. 78).

403. E. S. Wright: Using DECIPHER v2.0 to Analyze Big Biological Sequence Data in R. *The R Journal* **8**(1) (May 1, 2016), 352–359. doi: 10.32614/rj-2016-025 (cit. on p. 78).

404. M. Zamani and S. C. Kremer: Amino Acid Encoding Schemes for Machine Learning Methods. *2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW)*. Nov. 2011, 327–333. doi: 10.1109/bibmw.2011.6112394 (cit. on pp. 78–80).

405. D. Singh, P. Singh, and D. S. Sisodia: Evolutionary Based Optimal Ensemble Classifiers for HIV-1 Protease Cleavage Sites Prediction. *Expert Systems with Applications* **109** (Nov. 1, 2018), 86–99. doi: 10.1016/j.eswa.2018.05.003 (cit. on pp. 78, 80).

406. N. Qian and T. J. Sejnowski: Predicting the Secondary Structure of Globular Proteins Using Neural Network Models. *Journal of Molecular Biology* **202**(4) (Aug. 20, 1988), 865–884. doi: 10.1016/0022-2836(88)90564-5 (cit. on p. 78).

407. S. Budach and A. Marsico: Pysster: Classification of Biological Sequences by Learning Sequence and Structure Motifs with Convolutional Neural Networks. *Bioinformatics* **34**(17) (Sept. 1, 2018), 3035–3037. doi: 10.1093/bioinformatics/bty222 (cit. on p. 78).

408. A. C. H. Choong and N. K. Lee: Evaluation of Convolutionary Neural Networks Modeling of DNA Sequences Using Ordinal versus One-Hot Encoding Method. *2017 International Conference on Computer and Drone Applications (IConDA)*. Nov. 2017, 60–65. doi: 10.1109/iconda.2017.8270400 (cit. on p. 78).

409. W. McGinnis et al.: *Scikit-Learn-Contrib/Categorical-Encoding: Release For Zenodo*. Zenodo. Zenodo, Jan. 22, 2018. doi: 10.5281/zenodo.1157110 (cit. on pp. 79, 158).

410. S. Kawashima et al.: AAindex: amino acid index database, progress report 2008. *Nucleic Acids Research* **36**(suppl_1) (suppl_1 Jan. 1, 2008), D202–d205. doi: 10.1093/nar/gkm998 (cit. on pp. 79, 165).

411. Z.-C. Li, X.-B. Zhou, Z. Dai, and X.-Y. Zou: Prediction of Protein Structural Classes by Chou's Pseudo Amino Acid Composition: Approached Using Continuous Wavelet Transform and Principal Component Analysis. *Amino Acids* **37**(2) (Aug. 23, 2008), 415. doi: 10.1007/s00726-008-0170-2 (cit. on p. 79).

412. L. Nanni and A. Lumini: A New Encoding Technique for Peptide Classification. *Expert Systems with Applications* **38**(4) (Apr. 1, 2011), 3185–3191. doi: 10.1016/j.eswa.2010.09.005 (cit. on p. 79).

413. Z. Chen et al.: iFeature: A Python Package and Web Server for Features Extraction and Selection from Protein and Peptide Sequences. *Bioinformatics* **34**(14) (July 15, 2018), 2499–2502. doi: 10.1093/bioinformatics/bty140 (cit. on p. 79).

414. W. R. Taylor: The Classification of Amino Acid Conservation. *Journal of Theoretical Biology* **119**(2) (Mar. 21, 1986), 205–218. doi: 10.1016/s0022-5193(86)80075-3 (cit. on p. 79).

415. M. J. Zvelebil, G. J. Barton, W. R. Taylor, and M. J. E. Sternberg: Prediction of Protein Secondary Structure and Active Sites Using the Alignment of Homologous Sequences. *Journal of Molecular Biology* **195**(4) (June 20, 1987), 957–961. doi: 10.1016/0022-2836(87)90501-8 (cit. on p. 80).

416. S. Kremer and H. Lac: Method, System and Computer Program Product for Levinthal Process Induction from Known Structure Using Machine Learning. U.S. pat. 20090024375a1. University of Guelph. Jan. 22, 2009. https://patents.google.com/patent/US20090024375A1/en (cit. on p. 80).

417. S. Maetschke, M. Towsey, and M. Bodén: Blomap: An Encoding of Amino Acids Which Improves Signal Peptide Cleavage Site Prediction. *Proceedings of the 3rd Asia-Pacific Bioinformatics Conference*. **Volume 1**. 0 vols. Volume 1. Jan. 1, 2005, 141–150. doi: 10.1142/9781860947322_0014 (cit. on p. 80).

418. M. Gök and A. T. Özcerit: A New Feature Encoding Scheme for HIV-1 Protease Cleavage Site Prediction. *Neural Computing and Applications* **22**(7) (June 1, 2013), 1757–1761. doi: 10.1007/s00521-012-0967-5 (cit. on p. 80).

419. S. Saha and T. Bhattacharya: A Novel Approach to Find the Saturation Point of N-Gram Encoding Method for Protein Sequence Classification Involving Data Mining. *International Conference on Innovative Computing and Communications*. Singapore, 2019, 101–108. doi: 10.1007/978-981-13-2354-6_12 (cit. on p. 80).

420. H. Jeffrey: Chaos Game Representation of Gene Structure. *Nucleic Acids Research* **18**(8) (Apr. 25, 1990), 2163–2170. doi: 10.1093/nar/18.8.2163 (cit. on p. 80).

421. H. F. Löchel and D. Heider: Chaos Game Representation and Its Applications in Bioinformatics. *Computational and Structural Biotechnology Journal* **19** (Jan. 1, 2021), 6263–6271. doi: 10.1016/j.csbj.2021.11.008 (cit. on p. 80).

422. J. A. Cartes, S. Anand, S. Ciccolella, P. Bonizzoni, and G. D. Vedova: *Accurate and Fast Clade Assignment via Deep Learning and Frequency Chaos Game Representation*. June 13, 2022. doi: 10.1101/2022.06.13.495912 (cit. on p. 80).

423. H. Ni, H. Mu, and D. Qi: Applying Frequency Chaos Game Representation with Perceptual Image Hashing to Gene Sequence Phylogenetic Analyses. *Journal of Molecular Graphics and Modelling* **107** (Sept. 1, 2021), 107942. doi: 10.1016/j.jmgm.2021.107942 (cit. on p. 80).

424. A. Lwoff: The Concept of Virus. *Journal of General Microbiology* **17**(2) (Oct. 1957), 239–253. doi: 10.1099/00221287-17-2-239 (cit. on p. 83).

425. P. D. Minor: Viruses. *eLS*. 2014. doi: 10.1002/9780470015902.a0000441.pub3 (cit. on p. 83).

426. J. T. Stapleton, S. Foung, A. S. Muerhoff, J. Bukh, and P. Simmonds: The GB Viruses: A Review and Proposed Classification of GBV-A, GBV-C (HGV), and GBV-D in Genus Pegivirus within the Family Flaviviridae. *The Journal of General Virology* **92** (Pt 2 Feb. 2011), 233–246. doi: 10.1099/vir.0.027490-0 (cit. on p. 83).

427. N. Yamamoto et al.: Characterization of a Non-Pathogenic H5N1 Influenza Virus Isolated from a Migratory Duck Flying from Siberia in Hokkaido, Japan, in October 2009. *Virology Journal* **8**(1) (Feb. 11, 2011), 65. doi: 10.1186/1743-422x-8-65 (cit. on p. 83).

428. M. Shi et al.: The Evolutionary History of Vertebrate RNA Viruses. *Nature* **556**(7700) (7700 Apr. 2018), 197–202. doi: 10.1038/s41586-018-0012-7 (cit. on p. 83).

429. J. R. Adams and J.-R. Bonami: *Atlas of Invertebrate Viruses*. Boca Raton, Aug. 31, 2017. doi: 10.1201/9781315149929 (cit. on p. 83).

430. P. Lefeuvre et al.: Evolution and Ecology of Plant Viruses. *Nature Reviews Microbiology* **17**(10) (10 Oct. 2019), 632–644. doi: 10.1038/s41579-019-0232-3 (cit. on p. 83).

431. A. L. Wang and C. C. Wang: Viruses of Parasitic Protozoa. *Parasitology Today* **7**(4) (Jan. 1, 1991), 76–80. doi: 10.1016/0169-4758(91)90198-w (cit. on p. 83).

432. G. Fermin, S. Mazumdar-Leighton, and P. Tennant: Viruses of Prokaryotes, Protozoa, Fungi, and Chromista. *Viruses: Molecular Biology, Host Interactions, and Applications to Biotechnology*. 2018, 217. doi: 10.1016/B978-0-12-811257-1.00009-7 (cit. on p. 83).

433. S. Sutela, A. Poimala, and E. J. Vainio: Viruses of Fungi and Oomycetes in the Soil Environment. *FEMS Microbiology Ecology* **95**(9) (Sept. 1, 2019), fiz119. doi: 10.1093/femsec/fiz119 (cit. on p. 83).

434. F. Twort: An Investigation On The Nature Of Ultra-microscopic Viruses. *The Lancet* **186**(4814) (Dec. 1915), 1241–1243. doi: 10.1016/s0140-6736(01)20383-3 (cit. on p. 83).

435. M. Delbrock: Bacterial Viruses or Bacteriophages. *Biological Reviews* **21**(1) (1946), 30–40. doi: 10.1111/j.1469-185X.1946.tb00451.x (cit. on p. 83).

436. J. R. Clark and J. B. March: Bacterial Viruses as Human Vaccines? *Expert Review of Vaccines* **3**(4) (Aug. 1, 2004), 463–476. doi: 10.1586/14760584.3.4.463 (cit. on p. 83).

437. H. E. van Kan-Davelaar, J. C. M. van Hest, J. J. L. M. Cornelissen, and M. S. T. Koay: Using Viruses as Nanomedicines. *British Journal of Pharmacology* **171**(17) (2014), 4001–4009. doi: 10.1111/bph.12662 (cit. on p. 83).

438. D. Prangishvili, T. Basta, R. A. Garrett, and M. Krupovic: Viruses of the Archaea. *eLS*. 2016, 1–9. doi: 10.1002/9780470015902.a0000774.pub3 (cit. on p. 83).

439. D. Prangishvili, P. Forterre, and R. A. Garrett: Viruses of the Archaea: A Unifying View. *Nature Reviews Microbiology* **4**(11) (11 Nov. 2006), 837–848. doi: 10.1038/nrmicro1527 (cit. on p. 83).

440. R. Francki: Plant Virus Satellites. *Annual Review Of Microbiology* (Research 1985). 10.1146/annurev.mi.39.100185.001055 (cit. on p. 83).

441. P. Xu and M. J. Roossinck: Plant Virus Satellites. *eLS*. 2011. doi: 10.1002/9780470015902.a0000771.pub2 (cit. on p. 83).

442. M. M. Lai: The Molecular Biology of Hepatitis Delta Virus. *Annual review of biochemistry* **64** (Jan. 1, 1995), 259–286. doi: 10.1146/annurev.bi.64.070195.001355 (cit. on p. 83).

443. S. A. Hughes, H. Wedemeyer, and P. M. Harrison: Hepatitis Delta Virus. *The Lancet* **378**(9785) (July 2, 2011), 73–85. doi: 10.1016/s0140-6736(10)61931-9 (cit. on p. 83).

444. C. Desnues, M. Boyer, and D. Raoult: Chapter 3 - Sputnik, a Virophage Infecting the Viral Domain of Life. *Advances in Virus Research*. **82**. Jan. 1, 2012, 63–89. doi: 10.1016/b978-0-12-394621-8.00013-3 (cit. on p. 83).

445. M. Gaia et al.: Zamilon, a Novel Virophage with Mimiviridae Host Specificity. *Plos One* **9**(4) (Apr. 18, 2014), e94923. doi: 10.1371/journal.pone.0094923 (cit. on p. 83).

446. R. C. Edgar et al.: Petabase-Scale Sequence Alignment Catalyses Viral Discovery. *Nature* **602**(7895) (7895 Feb. 2022), 142–147. doi: 10.1038/s41586-021-04332-2 (cit. on p. 83).

447. A. Nasir, E. Romero-Severson, and J.-M. Claverie: Investigating the Concept and Origin of Viruses. *Trends in Microbiology* **28**(12) (Dec. 1, 2020), 959–967. doi: 10.1016/j.tim.2020.08.003 (cit. on p. 83).

448. P. Forterre and D. Prangishvili: The Origin of Viruses. *Research in Microbiology* **160**(7) (Sept. 1, 2009), 466–472. doi: 10.1016/j.resmic.2009.07.008 (cit. on p. 83).

449. P. Forterre: The Origin of Viruses and Their Possible Roles in Major Evolutionary Transitions. *Virus Research* **117**(1) (Apr. 1, 2006), 5–16. doi: 10.1016/j.virusres.2006.01.010 (cit. on p. 83).

450. J. Boeke and J. Stoye: Retrotransposons, Endogenous Retroviruses, and the Evolution of Retroelement. *Retroviruses*. Cold Spring Harbor (NY), 1997. https://www.ncbi.nlm.nih.gov/books/NBK19468 (cit. on p. 84).

451. S. Kojima et al.: Virus-like Insertions with Sequence Signatures Similar to Those of Endogenous Nonretroviral RNA Viruses in the Human Genome. *Proceedings of the National Academy of Sciences* **118**(5) (Feb. 2, 2021), e2010758118. doi: 10.1073/pnas.2010758118 (cit. on p. 84).

452. R. Löwer, J. Löwer, and R Kurth: The Viruses in All of Us: Characteristics and Biological Significance of Human Endogenous Retrovirus Sequences. *Proceedings of the National Academy of Sciences* **93**(11) (1996), 5177–5184. doi: 10.1073/pnas.93.11.5177 (cit. on p. 84).

453. D. J. Griffiths: Endogenous Retroviruses in the Human Genome Sequence. *Genome Biology* **2**(6) (June 5, 2001), reviews1017.1. doi: 10.1186/gb-2001-2-6-reviews1017 (cit. on p. 84).

454. D. Baltimore: Expression of Animal Virus Genomes. *Bacteriological Reviews* **35**(3) (Sept. 1971), 235–241. doi: 10.1128/br.35.3.235-241.1971 (cit. on p. 84).

455. E. V. Koonin, M. Krupovic, and V. I. Agol: The Baltimore Classification of Viruses 50 Years Later: How Does It Stand in the Light of Virus Evolution? *Microbiology and Molecular Biology Reviews* **85**(3) (Aug. 18, 2021), e00053–21. doi: 10.1128/mmbr.00053-21 (cit. on p. 84).

456. E. Domingo and C. Perales: RNA Virus Genomes. *eLS*. 2018, 1–12. doi: 10.1002/9780470015902.a0001488.pub3 (cit. on p. 84).

457. D. J. McGeoch, F. J. Rixon, and A. J. Davison: Topics in Herpesvirus Genomics and Evolution. *Virus Research* **117**(1) (Apr. 1, 2006), 90–104. doi: 10.1016/j.virusres.2006.01.002 (cit. on p. 84).

458. P. Boehmer and A. Nimonkar: Herpes Virus Replication. *IUBMB Life* **55**(1) (2003), 13–22. doi: 10.1080/1521654031000070645 (cit. on p. 84).

459. M. H. Brentjens, K. A. Yeung-Yue, P. C. Lee, and S. K. Tyring: Human Papillomavirus: A Review. *Dermatologic Clinics* **20**(2) (Apr. 1, 2002), 315–331. doi: 10.1016/s0733-8635(01)00028-6 (cit. on p. 84).

460. A. Kay and F. Zoulim: Hepatitis B Virus Genetic Variability and Evolution. *Virus Research* **127**(2) (Aug. 1, 2007), 164–176. doi: 10.1016/j.virusres.2007.02.021 (cit. on p. 84).

461. U. D. Parashar, J. S. Bresee, J. R. Gentsch, and R. I. Glass: Rotavirus. *Emerging Infectious Diseases* **4**(4) (1998), 561–570. doi: 10.3201/eid0404.980406 (cit. on p. 84).

462. P. Simmonds: Variability of Hepatitis C Virus. *Hepatology* **21**(2) (Feb. 1, 1995), 570–583. doi: 10.1016/0270-9139(95)90121-3 (cit. on p. 84).

463. E. Wimmer, C. U. T. Hellen, and X. Cao: Genetics of Poliovirus. *Annual Review of Genetics* **27** (Jan. 1, 1993), 353–437. doi: 10.1146/annurev.ge.27.120193.002033 (cit. on p. 84).

464. V. R. Racaniello: One Hundred Years of Poliovirus Pathogenesis. *Virology* **344**(1) (Jan. 5, 2006), 9–16. doi: 10.1016/j.virol.2005.09.015 (cit. on p. 84).

465. P. Palese, H. Zheng, O. G. Engelhardt, S. Pleschka, and A. García-Sastre: Negative-Strand RNA Viruses: Genetic Engineering and Applications. *Proceedings of the National Academy of Sciences* **93**(21) (Oct. 15, 1996), 11354–11358. doi: 10.1073/pnas.93.21.11354 (cit. on p. 84).

466. E. Domingo and C. Perales: Virus Evolution. *eLS*. 2014. doi: 10.1002/9780470015902.a0000436.pub3 (cit. on p. 84).

467. P. V'kovski, A. Kratzel, S. Steiner, H. Stalder, and V. Thiel: Coronavirus Biology and Replication: Implications for SARS-CoV-2. *Nature Reviews Microbiology* **19**(3) (3 Mar. 2021), 155–170. doi: 10.1038/s41579-020-00468-6 (cit. on p. 84).

468. A. T. Bäck and Å. Lundkvist: Dengue Viruses – an Overview. *Infection Ecology & Epidemiology* **3** (Aug. 30, 2013), 10.3402/iee.v3i0.19839. doi: 10.3402/iee.v3i0.19839 (cit. on p. 84).

469. L. B. Dustin, B. Bartolini, M. R. Capobianchi, and M. Pistello: Hepatitis C Virus: Life Cycle in Cells, Infection and Host Response, and Analysis of Molecular Markers Influencing the Outcome of Infection and Response to Therapy. *Clinical microbiology and infection : the official publication of the European Society of Clinical Microbiology and Infectious Diseases* **22**(10) (Oct. 2016), 826–832. doi: 10.1016/j.cmi.2016.08.025 (cit. on p. 84).

470. M. Kadaja, T. Silla, E. Ustav, and M. Ustav: Papillomavirus DNA Replication — From Initiation to Genomic Instability. *Virology* **384**(2) (Feb. 20, 2009), 360–368. doi: 10.1016/j.virol.2008.11.032 (cit. on p. 84).

471. S. K. Weller and D. M. Coen: Herpes Simplex Viruses: Mechanisms of DNA Replication. *Cold Spring Harbor Perspectives in Biology* **4**(9) (Sept. 2012), a013011. doi: 10.1101/cshperspect.a013011 (cit. on p. 84).

472. J. Beck and M. Nassal: Hepatitis B Virus Replication. *World Journal of Gastroenterology : WJG* **13**(1) (Jan. 7, 2007), 48–64. doi: 10.3748/wjg.v13.i1.48 (cit. on p. 84).

473. J. D. Pyle and K.-B. G. Scholthof: Chapter 58 - Biology and Pathogenesis of Satellite Viruses. *Viroids and Satellites*. Boston, Jan. 1, 2017, 627–636. doi: 10.1016/b978-0-12-801498-1.00058-9 (cit. on p. 84).

474. D. Raoult et al.: The 1.2-Megabase Genome Sequence of Mimivirus. *Science* **306**(5700) (Nov. 19, 2004), 1344–1350. doi: 10.1126/science.1101485 (cit. on p. 84).

475. J. A. Campillo-Balderas, A. Lazcano, and A. Becerra: Viral Genome Size Distribution Does Not Correlate with the Antiquity of the Host Lineages. *Frontiers in Ecology and Evolution* **3** (2015). doi: 10.3389/fevo.2015.00143 (cit. on p. 84).

476. A. J. Cann: Virus Structure. *eLS*. 2015, 1–9. doi: 10.1002/9780470015902.a0000439.pub2 (cit. on p. 84).

477. F. Hladik and M. J. McElrath: Setting the Stage: Host Invasion by HIV. *Nature Reviews Immunology* **8**(6) (6 June 2008), 447–457. doi: 10.1038/nri2302 (cit. on p. 85).

478. G. M. Shaw and E. Hunter: HIV Transmission. *Cold Spring Harbor Perspectives in Medicine* **2**(11) (Jan. 11, 2012), a006965. doi: 10.1101/cshperspect.a006965 (cit. on p. 85).

479. R. A. Weiss: How Does HIV Cause AIDS? *Science* **260**(5112) (May 28, 1993), 1273–1279. doi: 10.1126/science.8493571 (cit. on p. 85).

480. A. Melhuish and P. Lewthwaite: Natural History of HIV and AIDS. *Medicine* **46**(6) (June 1, 2018), 356–361. doi: 10.1016/j.mpmed.2018.03.010 (cit. on p. 85).

481. J. F. Murray et al.: Pulmonary Complications of the Acquired Immunodeficiency Syndrome. *New England Journal of Medicine* **310**(25) (1984), 1682–1688. doi: 10.1056/nejm198406213102529 (cit. on p. 85).

482. S. Sampath et al.: Pandemics Throughout the History. *Cureus* **13**(9) (Sept. 20, 2021). doi: 10.7759/cureus.18136 (cit. on p. 85).

483. World Health Organization: *Global Report: UNAIDS Report on the Global AIDS Epidemic 2010.* Geneve, 2010. https://www.unaids.org/globalreport/Global_report.htm (cit. on p. 85).

484. F. Barré-Sinoussi et al.: Isolation of a T-lymphotropic Retrovirus from a Patient at Risk for Acquired Immune Deficiency Syndrome (AIDS). *Science* **220**(4599) (May 20, 1983), 868–871. doi: 10.1126/science.6189183 (cit. on p. 85).

485. R. C. Gallo et al.: Isolation of Human T-cell Leukemia Virus in Acquired Immune Deficiency Syndrome (AIDS). *Science* **220**(4599) (May 20, 1983), 865–867. doi: 10.1126/science.6601823 (cit. on p. 85).

486. F. Clavel et al.: Isolation of a New Human Retrovirus from West African Patients with AIDS. *Science* **233**(4761) (July 18, 1986), 343–346. doi: 10.1126/science.2425430 (cit. on p. 85).

487. P. B. Gilbert et al.: Comparison of HIV-1 and HIV-2 Infectivity from a Prospective Cohort Study in Senegal. *Statistics in Medicine* **22**(4) (Feb. 28, 2003), 573–593. doi: 10.1002/sim.1342 (cit. on p. 85).

488. M. F. S. van der Loeff et al.: Sixteen Years of HIV Surveillance in a West African Research Clinic Reveals Divergent Epidemic Trends of HIV-1 and HIV-2. *International Journal of Epidemiology* **35**(5) (Oct. 2006), 1322–1328. doi: 10.1093/ije/dyl037 (cit. on p. 85).

489. F. Gao et al.: Origin of HIV-1 in the Chimpanzee Pan Troglodytes Troglodytes. *Nature* **397**(6718) (6718 Feb. 1999), 436–441. doi: 10.1038/17130 (cit. on p. 85).

490. D. J. Hamel et al.: Twenty Years of Prospective Molecular Epidemiology in Senegal: Changes in HIV Diversity. *AIDS research and human retroviruses* **23**(10) (Oct. 2007), 1189–1196. doi: 10.1089/aid.2007.0037 (cit. on p. 85).

491. P. M. Sharp and B. H. Hahn: Origins of HIV and the AIDS Pandemic. *Cold Spring Harbor Perspectives in Medicine:* **1**(1) (Sept. 2011), a006841. doi: 10.1101/cshperspect.a006841 (cit. on pp. 85, 86).

492. V. M. Hirsch, R. A. Olmsted, M. Murphey-Corb, R. H. Purcell, and P. R. Johnson: An African Primate Lentivirus (SIVsmclosely Related to HIV-2. *Nature* **339**(6223) (6223 June 1989), 389–392. doi: 10.1038/339389a0 (cit. on p. 85).

493. F. Gao et al.: Human Infection by Genetically Diverse SIVSM-related HIV-2 in West Africa. *Nature* **358**(6386) (6386 Aug. 1992), 495–499. doi: 10.1038/358495a0 (cit. on p. 85).

494. Z. Chen et al.: Genetic Characterization of New West African Simian Immunodeficiency Virus SIVsm: Geographic Clustering of Household-Derived SIV Strains with Human Immunodeficiency Virus Type 2 Subtypes and Genetically Diverse Viruses from a Single Feral Sooty Mangabey Troop. *Journal of Virology* **70**(6) (June 1996), 3617–3627. doi: 10.1128/jvi.70.6.3617-3627.1996 (cit. on p. 85).

495. J. Hemelaar: The Origin and Diversity of the HIV-1 Pandemic. *Trends in Molecular Medicine* **18**(3) (Mar. 1, 2012), 182–192. doi: 10.1016/j.molmed.2011.12.001 (cit. on p. 85).

496. M. Worobey et al.: Direct Evidence of Extensive Diversity of HIV-1 in Kinshasa by 1960. *Nature* **455**(7213) (7213 Oct. 2008), 661–664. doi: 10.1038/nature07390 (cit. on p. 86).

497. N. Vidal et al.: Unprecedented Degree of Human Immunodeficiency Virus Type 1 (HIV-1) Group M Genetic Diversity in the Democratic Republic of Congo Suggests That the HIV-1 Pandemic Originated in Central Africa. *Journal of Virology* **74**(22) (Nov. 2000), 10498–10507. doi: 10.1128/jvi.74.22.10498-10507.2000 (cit. on p. 86).

498. N. R. Faria et al.: The Early Spread and Epidemic Ignition of HIV-1 in Human Populations. *Science* **346**(6205) (Oct. 3, 2014), 56–61. doi: 10.1126/science.1256739 (cit. on p. 86).

499. B. Korber et al.: Timing the Ancestor of the HIV-1 Pandemic Strains. *Science* **288**(5472) (June 9, 2000), 1789–1796. doi: 10.1126/science.288.5472.1789 (cit. on p. 86).

500. A. Rambaut, D. Posada, K. A. Crandall, and E. C. Holmes: The Causes and Consequences of HIV Evolution. *Nature Reviews Genetics* **5**(1) (1 Jan. 2004), 52–61. doi: 10.1038/nrg1246 (cit. on p. 86).

501. F. E. McCutchan: Global Epidemiology of HIV. *Journal of Medical Virology* **78**(S1) (2006), S7–s12. doi: 10.1002/jmv.20599 (cit. on p. 86).

502. M. Pérez-Losada, M. Arenas, J. C. Galán, F. Palero, and F. González-Candelas: Recombination in Viruses: Mechanisms, Methods of Study, and Evolutionary Consequences. *Infection, Genetics and Evolution* **30** (Mar. 2015), 296–307. doi: 10.1016/j.meegid.2014.12.022 (cit. on p. 86).

503. D. L. Robertson, B. H. Hahn, and P. M. Sharp: Recombination in AIDS Viruses. *Journal of Molecular Evolution* **40**(3) (Mar. 1995), 249–259. doi: 10.1007/bf00163230 (cit. on p. 86).

504. *HIV Circulating Recombinant Forms (CRFs)*. Los Alamos National Laboratory. https://www.hiv.lanl.gov/content/sequence/HIV/CRFs/CRFs.html (cit. on p. 86).

505. K. A. Lau and J. J. Wong: Current Trends of HIV Recombination Worldwide. *Infectious Disease Reports* **5** (Suppl 1 June 6, 2013), e4. doi: 10.4081/idr.2013.s1.e4 (cit. on p. 86).

506. D. Posada, K. A. Crandall, and E. C. Holmes: Recombination in Evolutionary Genomics. *Annual Review of Genetics* **36** (2002), 75–97. doi: 10.1146/annurev.genet.36.040202.111115 (cit. on p. 86).

507. B. S. Taylor, M. E. Sobieszczyk, F. E. McCutchan, and S. M. Hammer: The Challenge of HIV-1 Subtype Diversity. *New England Journal of Medicine* **358**(15) (Apr. 10, 2008), 1590–1602. doi: 10.1056/NEJMra0706737 (cit. on p. 86).

508. J. Hemelaar, E. Gouws, P. D. Ghys, and S. Osmanov: Global Trends in Molecular Epidemiology of HIV-1 during 2000–2007. *AIDS (London, England)* **25**(5) (Mar. 13, 2011), 679–689. doi: 10.1097/QAD.0b013e328342ff93 (cit. on p. 86).

509. *Distribution of All HIV-1 Sequences: WORLD*. Los Alamos National Laboratory. https://www.hiv.lanl.gov/components/sequence/HIV/geo/geo.comp (cit. on p. 86).

510. E. O. Freed: HIV-1 Replication. *Somatic Cell and Molecular Genetics* **26**(1) (Nov. 1, 2001), 13–33. doi: 10.1023/a:1021070512287 (cit. on p. 86).

511. M. R. Ferguson, D. R. Rojo, J. J. von Lindern, and W. A. O'Brien: HIV-1 Replication Cycle. *Clinics in Laboratory Medicine* **22**(3) (Sept. 2002), 611–635. doi: 10.1016/s0272-2712(02)00015-x (cit. on p. 86).

512. M. L. Gougeon, A. G. Laurent-Crawford, A. G. Hovanessian, and L. Montagnier: Direct and Indirect Mechanisms Mediating Apoptosis during HIV Infection: Contribution to in Vivo CD4 T Cell Depletion. *Seminars in Immunology* **5**(3) (June 1, 1993), 187–194. doi: 10.1006/smim.1993.1022 (cit. on p. 87).

513. K. K. Vidya Vijayan, K. P. Karthigeyan, S. P. Tripathi, and L. E. Hanna: Pathophysiology of CD4+ T-Cell Depletion in HIV-1 and HIV-2 Infections. *Frontiers in Immunology* **8** (May 23, 2017), 580. doi: 10.3389/fimmu.2017.00580 (cit. on p. 87).

514. A. D. Frankel and J. A. Young: HIV-1: Fifteen Proteins and an RNA. *Annual Review of Biochemistry* **67** (1998), 1–25. doi: 10.1146/annurev.biochem.67.1.1 (cit. on p. 87).

515. T. Fossen et al.: Solution Structure of the Human Immunodeficiency Virus Type 1 P6 Protein *. *Journal of Biological Chemistry* **280**(52) (Dec. 30, 2005), 42515–42527. doi: 10.1074/jbc.M507375200 (cit. on p. 88).

516. H. G. Göttlinger, T. Dorfman, J. G. Sodroski, and W. A. Haseltine: Effect of Mutations Affecting the P6 Gag Protein on Human Immunodeficiency Virus Particle Release. *Proceedings of the National Academy of Sciences* **88**(8) (Apr. 15, 1991), 3195–3199. doi: 10.1073/pnas.88.8.3195 (cit. on p. 88).

517. M. Huang, J. M. Orenstein, M. A. Martin, and E. O. Freed: p6Gag Is Required for Particle Production from Full-Length Human Immunodeficiency Virus Type 1 Molecular Clones Expressing Protease. *Journal of Virology* **69**(11) (Nov. 1995), 6810–6818. doi: 10.1128/jvi.69.11.6810-6818.1995 (cit. on p. 88).

518. S. Bour, R. Geleziunas, and M. A. Wainberg: The Human Immunodeficiency Virus Type 1 (HIV-1) CD4 Receptor and Its Central Role in Promotion of HIV-1 Infection. *Microbiological Reviews* **59**(1) (Mar. 1995), 63–93. doi: 10.1128/mr.59.1.63-93.1995 (cit. on p. 88).

519. L. D. Hernandez, L. R. Hoffman, T. G. Wolfsberg, and J. M. White: Virus-Cell and Cell-Cell Fusion. *Annual Review of Cell and Developmental Biology* **12** (1996), 627–661. doi: 10.1146/annurev.cellbio.12.1.627 (cit. on p. 88).

520. K. Jones and B. Peterlin: Control of Rna Initiation and Elongation at the Hiv-1 Promoter. *Annual Review of Biochemistry* **63** (1994), 717–743. doi: 10.1146/annurev.bi.63.070194.003441 (cit. on p. 89).

521. T. J. Hope: Viral RNA Export. *Chemistry & Biology* **4**(5) (May 1, 1997), 335–344. doi: 10.1016/s1074-5521(97)90124-1 (cit. on p. 89).

522. A. Mangasarian and D. Trono: The Multifaceted Role of HIV Nef. *Research in Virology* **148**(1) (Jan. 1, 1997), 30–33. doi: 10.1016/s0923-2516(97)81909-7 (cit. on p. 89).

523. é. A. Cohen, R. A. Subbramanian, and H. G. Göttlinger: Role of Auxiliary Proteins in Retroviral Morphogenesis. *Morphogenesis and Maturation of Retroviruses*. Berlin, Heidelberg, 1996, 219–235. doi: 10.1007/978-3-642-80145-7_7 (cit. on p. 89).

524. R. A. Lamb and L. H. Pinto: Do Vpu and Vpr of Human Immunodeficiency Virus Type 1 and NB of Influenza B Virus Have Ion Channel Activities in the Viral Life Cycles? *Virology* **229**(1) (Mar. 3, 1997), 1–11. doi: 10.1006/viro.1997.8451 (cit. on p. 89).

525. N. Khan and J. D. Geiger: Role of Viral Protein U (Vpu) in HIV-1 Infection and Pathogenesis. *Viruses* **13**(8) (July 27, 2021), 1466. doi: 10.3390/v13081466 (cit. on p. 89).

526. M. Emerman: HIV-1, Vpr and the Cell Cycle. *Current Biology* **6**(9) (Sept. 1, 1996), 1096–1103. doi: 10.1016/s0960-9822(02)00676-0 (cit. on p. 89).

527. R. H. Miller: Human Immunodeficiency Virus May Encode a Novel Protein on the Genomic DNA plus Strand. *Science* **239**(4846) (Mar. 18, 1988), 1420–1422. doi: 10.1126/science.3347840 (cit. on p. 89).

528. S. Briquet and C. Vaquero: Immunolocalization Studies of an Antisense Protein in HIV-1-Infected Cells and Viral Particles. *Virology* **292**(2) (Jan. 20, 2002), 177–184. doi: 10.1006/viro.2001.1224 (cit. on p. 89).

529. E. Cassan, A.-M. Arigon-Chifolleau, J.-M. Mesnard, A. Gross, and O. Gascuel: Concomitant Emergence of the Antisense Protein Gene of HIV-1 and of the Pandemic. *Proceedings of the National Academy of Sciences* **113**(41) (Oct. 11, 2016), 11537–11542. doi: 10.1073/pnas.1605739113 (cit. on p. 89).

530. J. Savoret et al.: A Pilot Study of the Humoral Response Against the AntiSense Protein (ASP) in HIV-1-Infected Patients. *Frontiers in Microbiology* **11** (2020). doi: 10.3389/fmicb.2020.00020 (cit. on p. 89).

531. C. Zardecki et al.: PDB-101: Educational Resources Supporting Molecular Explorations through Biology and Medicine. *Protein Science* **31**(1) (2022), 129–140. doi: 10.1002/pro.4200 (cit. on p. 90).

532. R. W. Eisinger, C. W. Dieffenbach, and A. S. Fauci: HIV Viral Load and Transmissibility of HIV Infection: Undetectable Equals Untransmittable. *Jama* **321**(5) (Feb. 5, 2019), 451–452. doi: 10.1001/jama.2018.21167 (cit. on p. 89).

533. F. J. Palella et al.: Declining Morbidity and Mortality among Patients with Advanced Human Immunodeficiency Virus Infection. *New England Journal of Medicine* **338**(13) (Mar. 26, 1998), 853–860. doi: 10.1056/nejm199803263381301 (cit. on p. 89).

534. S. S. Forsythe et al.: Twenty Years Of Antiretroviral Therapy For People Living With HIV: Global Costs, Health Achievements, Economic Benefits. *Health Affairs* **38**(7) (July 2019), 1163–1172. doi: 10.1377/hlthaff.2018.05391 (cit. on p. 89).

535. M. A. Fischl et al.: The Efficacy of Azidothymidine (AZT) in the Treatment of Patients with AIDS and AIDS-Related Complex. *New England Journal of Medicine* **317**(4) (July 23, 1987), 185–191. doi: 10.1056/nejm198707233170401 (cit. on p. 91).

536. D. D. Richman: Susceptibility to Nucleoside Analogues of Zidovudine-Resistant Isolates of Human Immunodeficiency Virus. *The American Journal of Medicine* **88** (5, Supplement 2 May 21, 1990), S8–s10. doi: 10.1016/0002-9343(90)90414-9 (cit. on p. 91).

537. J. Y. Yeo, G.-R. Goh, C. T.-T. Su, and S. K.-E. Gan: The Determination of HIV-1 RT Mutation Rate, Its Possible Allosteric Effects, and Its Implications on Drug Resistance. *Viruses* **12**(3) (3 Mar. 2020), 297. doi: 10.3390/v12030297 (cit. on p. 91).

538. J. M. Cuevas, R. Geller, R. Garijo, J. López-Aldeguer, and R. Sanjuán: Extremely High Mutation Rate of HIV-1 In Vivo. *PLOS Biology* **13**(9) (Sept. 16, 2015), e1002251. doi: 10.1371/journal.pbio.1002251 (cit. on p. 91).

539. A. Carvajal-Rodríguez, K. A. Crandall, and D. Posada: Recombination Favors the Evolution of Drug Resistance in HIV-1 during Antiretroviral Therapy. *Infection, genetics and evolution : journal of molecular epidemiology and evolutionary genetics in infectious diseases* **7**(4) (July 2007), 476–483. doi: 10.1016/j.meegid.2007.02.001 (cit. on p. 91).

540. R. M. Gulick et al.: Treatment with Indinavir, Zidovudine, and Lamivudine in Adults with Human Immunodeficiency Virus Infection and Prior Antiretroviral Therapy. *The New England Journal of Medicine* **337**(11) (Sept. 11, 1997), 734–739. doi: 10.1056/nejm199709113371102 (cit. on p. 91).

541. A. M. J. Wensing, N. M. van Maarseveen, and M. Nijhuis: Fifteen Years of HIV Protease Inhibitors: Raising the Barrier to Resistance. *Antiviral Research* **85**(1) (Jan. 1, 2010), 59–74. doi: 10.1016/j.antiviral.2009.10.003 (cit. on pp. 91, 96).

542. O. S. Pedersen and E. B. Pedersen: Non-Nucleoside Reverse Transcriptase Inhibitors: The NNRTI Boom. *Antiviral Chemistry and Chemotherapy* **10**(6) (Dec. 1, 1999), 285–314. doi: 10.1177/095632029901000601 (cit. on p. 91).

543. K. K. Scarsi, J. P. Havens, A. T. Podany, S. N. Avedissian, and C. V. Fletcher: HIV-1 Integrase Inhibitors: A Comparative Review of Efficacy and Safety. *Drugs* **80**(16) (Nov. 2020), 1649–1676. doi: 10.1007/s40265-020-01379-9 (cit. on p. 91).

544. C. V. Fletcher: Enfuvirtide, a New Drug for HIV Infection. *The Lancet* **361**(9369) (May 10, 2003), 1577–1578. doi: 10.1016/s0140-6736(03)13323-5 (cit. on p. 91).

545. J. A. Esté and A. Telenti: HIV Entry Inhibitors. *The Lancet* **370**(9581) (July 7, 2007), 81–88. doi: 10.1016/s0140-6736(07)61052-6 (cit. on p. 91).

546. J. M. Kilby and J. J. Eron: Novel Therapies Based on Mechanisms of HIV-1 Cell Entry. *New England Journal of Medicine* **348**(22) (May 29, 2003), 2228–2238. doi: 10.1056/NEJMra022812 (cit. on p. 91).

547. P. Yeni: Update on HAART in HIV. *Journal of Hepatology* **44** (Jan. 1, 2006), S100–s103. doi: 10.1016/j.jhep.2005.11.021 (cit. on p. 91).

548. L. Palmisano and S. Vella: A Brief History of Antiretroviral Therapy of HIV Infection: Success and Challenges. *Annali dell'Istituto Superiore Di Sanita* **47**(1) (2011), 44–48. doi: 10.4415/ann_11_01_10 (cit. on p. 91).

549. P. S. Pennings: HIV Drug Resistance: Problems and Perspectives. *Infectious Disease Reports* **5**(s1) (s1 June 2013), e5. doi: 10.4081/idr.2013.s1.e5 (cit. on p. 91).

550. S. Mehta, R. D. Moore, and N. M. H. Graham: Potential Factors Affecting Adherence with HIV Therapy. *Aids* **11**(14) (Nov. 15, 1997), 1665–1670. doi: 10.1097/00002030-199714000-00002 (cit. on p. 91).

551. N. H. Miller: Compliance with Treatment Regimens in Chronic Asymptomatic Diseases. *The American Journal of Medicine* **102** (2, Supplement 1 Feb. 17, 1997), 43–49. doi: 10.1016/s0002-9343(97)00467-1 (cit. on p. 91).

552. M. A. Chesney, M. Morin, and L. Sherr: Adherence to HIV Combination Therapy. *Social Science & Medicine* **50**(11) (June 1, 2000), 1599–1605. doi: 10.1016/s0277-9536(99)00468-2 (cit. on p. 91).

553. I. Aldir, A. Horta, and M. Serrado: Single-Tablet Regimens in HIV: Does It Really Make a Difference? *Current Medical Research and Opinion* **30**(1) (Jan. 1, 2014), 89–97. doi: 10.1185/03007995.2013.844685 (cit. on p. 92).

554. R. M. Grant et al.: Preexposure Chemoprophylaxis for HIV Prevention in Men Who Have Sex with Men. *New England Journal of Medicine* **363**(27) (Dec. 30, 2010), 2587–2599. doi: 10.1056/NEJMoa1011205 (cit. on p. 92).

555. J. M. Baeten et al.: Antiretroviral Prophylaxis for HIV Prevention in Heterosexual Men and Women. *The New England Journal of Medicine* **367**(5) (Aug. 2, 2012), 399–410. doi: 10.1056/NEJMoa1108524 (cit. on p. 92).

556. S. P. Buchbinder and A. Liu: Pre-Exposure Prophylaxis and the Promise of Combination Prevention Approaches. *AIDS and behavior* **15 Suppl 1** (Apr. 2011), S72–79. doi: 10.1007/s10461-011-9894-1 (cit. on p. 92).

557. J. Riddell IV, K. R. Amico, and K. H. Mayer: HIV Preexposure Prophylaxis: A Review. *Jama* **319**(12) (Mar. 27, 2018), 1261–1268. doi: 10.1001/jama.2018.1917 (cit. on p. 92).

558. *Truvada*. European Medicines Agency. Sept. 17, 2018. https://www.ema.europa.eu/en/medicines/human/EPAR/truvada (cit. on p. 92).

559. *About PrEP | PrEP | HIV Basics | HIV/AIDS*. Cdc. July 12, 2022. https://www.cdc.gov/hiv/basics/prep/about-prep.html (cit. on p. 92).

560. A. R. Zolopa: The Evolution of HIV Treatment Guidelines: Current State-of-the-Art of ART. *Antiviral Research* **85**(1) (Jan. 1, 2010), 241–244. doi: 10.1016/j.antiviral.2009.10.018 (cit. on p. 92).

561. W. H. Organization: *Consolidated Guidelines on HIV Prevention, Testing, Treatment, Service Delivery and Monitoring: Recommendations for a Public Health Approach*. World Health Organization, July 16, 2021. https://www.who.int/publications-detail-redirect/9789240031593 (cit. on p. 92).

562. P. Ammaranond and S. Sanguansittianan: Mechanism of HIV Antiretroviral Drugs Progress toward Drug Resistance. *Fundamental & Clinical Pharmacology* **26**(1) (2012), 146–161. doi: 10.1111/j.1472-8206.2011.01009.x (cit. on pp. 93, 95).

563. F. Clavel and A. J. Hance: HIV Drug Resistance. *New England Journal of Medicine* **350**(10) (Mar. 4, 2004), 1023–1035. doi: 10.1056/NEJMra025195 (cit. on p. 93).

564. S. G. Sarafianos et al.: Structure and Function of HIV-1 Reverse Transcriptase: Molecular Mechanisms of Polymerization and Inhibition. *Journal of molecular biology* **385**(3) (Jan. 23, 2009), 693–713. doi: 10.1016/j.jmb.2008.10.071 (cit. on pp. 93, 121).

565. D. S. Goodsell, L. Autin, and A. J. Olson: Illustrate: Software for Biomolecular Illustration. *Structure (London, England: 1993)* **27**(11) (Nov. 5, 2019), 1716–1720.e1. doi: 10.1016/j.str.2019.08.011 (cit. on pp. 94, 96).

566. R. M. Esnouf et al.: Unique Features in the Structure of the Complex between HIV-1 Reverse Transcriptase and the Bis(Heteroaryl)Piperazine (BHAP) U-90152 Explain Resistance Mutations for This Nonnucleoside Inhibitor. *Proceedings of the National Academy of Sciences of the United States of America* **94**(8) (Apr. 15, 1997), 3984–3989. doi: 10.1073/pnas.94.8.3984 (cit. on p. 95).

567. J. Q. Hang et al.: Activity of the Isolated HIV RNase H Domain and Specific Inhibition by N-hydroxyimides. *Biochemical and Biophysical Research Communications* **317**(2) (Apr. 30, 2004), 321–329. doi: 10.1016/j.bbrc.2004.03.061 (cit. on p. 95).

568. K. Klumpp and T. Mirzadegan: Recent Progress in the Design of Small Molecule Inhibitors of HIV RNase H. *Current Pharmaceutical Design* **12**(15) (May 1, 2006), 1909–1922. doi: 10.2174/138161206776873653 (cit. on p. 95).

569. L. Menéndez-Arias: Mechanisms of Resistance to Nucleoside Analogue Inhibitors of HIV-1 Reverse Transcriptase. *Virus Research* **134**(1) (June 1, 2008), 124–146. doi: 10.1016/j.virusres.2007.12.015 (cit. on p. 95).

570. N. Sluis-Cremer, D. Arion, and M. A. Parniak*: Molecular Mechanisms of HIV-1 Resistance to Nucleoside Reverse Transcriptase Inhibitors (NRTIs). *Cellular and Molecular Life Sciences CMLS* **57**(10) (Sept. 1, 2000), 1408–1422. doi: 10.1007/pl00000626 (cit. on p. 95).

571. S. G. Sarafianos et al.: Lamivudine (3TC) Resistance in HIV-1 Reverse Transcriptase Involves Steric Hindrance with Beta-Branched Amino Acids. *Proceedings of the National Academy of Sciences of the United States of America* **96**(18) (Aug. 31, 1999), 10027–10032. doi: 10.1073/pnas.96.18.10027 (cit. on p. 95).

572. P. R. Meyer, S. E. Matsuura, A. M. Mian, A. G. So, and W. A. Scott: A Mechanism of AZT Resistance: An Increase in Nucleotide-Dependent Primer Unblocking by Mutant HIV-1 Reverse Transcriptase. *Molecular Cell* **4**(1) (July 1999), 35–43. doi: 10.1016/s1097-2765(00)80185-9 (cit. on p. 95).

573. P. L. Boyer, S. G. Sarafianos, E. Arnold, and S. H. Hughes: Selective Excision of AZTMP by Drug-Resistant Human Immunodeficiency Virus Reverse Transcriptase. *Journal of Virology* **75**(10) (May 2001), 4832–4842. doi: 10.1128/jvi.75.10.4832-4842.2001 (cit. on p. 95).

574. S. G. Deeks: Nonnucleoside Reverse Transcriptase Inhibitor Resistance. *JAIDS Journal of Acquired Immune Deficiency Syndromes* **26** (Mar. 1, 2001), S25. doi: 10.1097/00126334-200103011-00004 (cit. on p. 95).

575. J. Ren and D. K. Stammers: Structural Basis for Drug Resistance Mechanisms for Non-Nucleoside Inhibitors of HIV Reverse Transcriptase. *Virus Research* **134**(1) (June 1, 2008), 157–170. doi: 10.1016/j.virusres.2007.12.018 (cit. on p. 95).

576. S. B. Lloyd, S. J. Kent, and W. R. Winnall: The High Cost of Fidelity. *AIDS Research and Human Retroviruses* **30**(1) (Jan. 2014), 8–16. doi: 10.1089/aid.2013.0153 (cit. on p. 95).

577. L. H. Pearl and W. R. Taylor: A Structural Model for the Retroviral Proteases. *Nature* **329**(6137) (6137 Sept. 1987), 351–354. doi: 10.1038/329351a0 (cit. on p. 95).

578. S. Gulnik, J. W. Erickson, and D. Xie: HIV Protease: Enzyme Function and Drug Resistance. *Vitamins & Hormones*. **58**. Jan. 1, 2000, 213–256. doi: 10.1016/s0083-6729(00)58026-1 (cit. on p. 95).

579. A. M. Silva, R. E. Cachau, H. L. Sham, and J. W. Erickson: Inhibition and Catalytic Mechanism of HIV-1 Aspartic Protease. *Journal of Molecular Biology* **255**(2) (Jan. 1, 1996), 321–340. doi: 10.1006/jmbi.1996.0026 (cit. on p. 95).

580. V. Hornak, A. Okur, R. C. Rizzo, and C. Simmerling: HIV-1 Protease Flaps Spontaneously Open and Reclose in Molecular Dynamics Simulations. *Proceedings of the National Academy of Sciences* **103**(4) (Jan. 24, 2006), 915–920. doi: 10.1073/pnas.0508452103 (cit. on p. 96).

581. D. I. Freedberg et al.: Rapid Structural Fluctuations of the Free HIV Protease Flaps in Solution: Relationship to Crystal Structures and Comparison with Predictions of Dynamics Calculations. *Protein Science : A Publication of the Protein Society* **11**(2) (Feb. 2002), 221–232. doi: 10.1110/ps.33202 (cit. on p. 96).

582. Y. Yu et al.: Structural Insights into HIV-1 Protease Flap Opening Processes and Key Intermediates. *RSC Advances* **7**(71) (Sept. 15, 2017), 45121–45128. doi: 10.1039/c7ra09691g (cit. on p. 96).

583. N. A. Roberts et al.: Rational Design of Peptide-Based HIV Proteinase Inhibitors. *Science* **248**(4953) (Apr. 20, 1990), 358–361. doi: 10.1126/science.2183354 (cit. on p. 96).

584. Z. Lv, Y. Chu, and Y. Wang: HIV Protease Inhibitors: A Review of Molecular Selectivity and Toxicity. *Hiv/aids* **7** (Apr. 8, 2015), 95–104. doi: 10.2147/hiv.s79956 (cit. on p. 96).

585. M. Prabu-Jeyabalan, E. Nalivaika, and C. A. Schiffer: Substrate Shape Determines Specificity of Recognition for HIV-1 Protease: Analysis of Crystal Structures of Six Substrate Complexes. *Structure* **10**(3) (Mar. 1, 2002), 369–381. doi: 10.1016/s0969-2126(02)00720-7 (cit. on p. 96).

586. M. Prabu-Jeyabalan et al.: Substrate Envelope and Drug Resistance: Crystal Structure of RO1 in Complex with Wild-Type Human Immunodeficiency Virus Type 1 Protease. *Antimicrobial Agents and Chemotherapy* **50**(4) (Apr. 2006), 1518–1521. doi: 10.1128/aac.50.4.1518-1521.2006 (cit. on p. 96).

587. N. Kurt Yilmaz, R. Swanstrom, and C. A. Schiffer: Improving Viral Protease Inhibitors to Counter Drug Resistance. *Trends in Microbiology* **24**(7) (July 1, 2016), 547–557. doi: 10.1016/j.tim.2016.03.010 (cit. on p. 97).

588. T. K. Chiu and D. R. Davies: Structure and Function of HIV-1 Integrase. *Current Topics in Medicinal Chemistry* **4**(9) (May 1, 2004), 965–977. doi: 10.2174/1568026043388547 (cit. on p. 97).

589. D. Esposito and R. Craigie: HIV Integrase Structure and Function. *Advances in Virus Research*. **52**. Jan. 1, 1999, 319–333. doi: 10.1016/s0065-3527(08)60304-8 (cit. on p. 97).

590. O. Delelis, K. Carayon, A. Saïb, E. Deprez, and J.-F. Mouscadet: Integrase and Integration: Biochemical Activities of HIV-1 Integrase. *Retrovirology* **5** (Dec. 17, 2008), 114. doi: 10.1186/1742-4690-5-114 (cit. on p. 97).

591. G. N. Maertens, A. N. Engelman, and P. Cherepanov: Structure and Function of Retroviral Integrase. *Nature Reviews Microbiology* **20**(1) (1 Jan. 2022), 20–34. doi: 10.1038/s41579-021-00586-9 (cit. on p. 98).

592. Y. Pommier, A. A. Johnson, and C. Marchand: Integrase Inhibitors to Treat HIV/Aids. *Nature Reviews Drug Discovery* **4**(3) (3 Mar. 2005), 236–248. doi: 10.1038/nrd1660 (cit. on p. 98).

593. J.-L. Blanco, V. Varghese, S.-Y. Rhee, J. M. Gatell, and R. W. Shafer: HIV-1 Integrase Inhibitor Resistance and Its Clinical Implications. *The Journal of Infectious Diseases* **203**(9) (May 1, 2011), 1204–1214. doi: 10.1093/infdis/jir025 (cit. on p. 98).

594. A. M. Geretti, D. Armenia, and F. Ceccherini-Silberstein: Emerging Patterns and Implications of HIV-1 Integrase Inhibitor Resistance. *Current Opinion in Infectious Diseases* **25**(6) (Dec. 2012), 677–686. doi: 10.1097/QCO.0b013e32835a1de7 (cit. on p. 98).

595. D. C. Knox, P. L. Anderson, P. R. Harrigan, and D. H. Tan: Multidrug-Resistant HIV-1 Infection despite Preexposure Prophylaxis. *New England Journal of Medicine* **376**(5) (Feb. 2, 2017), 501–502. doi: 10.1056/NEJMc1611639 (cit. on p. 98).

596. C. B. Hurt, J. J. Eron, and M. S. Cohen: Pre-Exposure Prophylaxis and Antiretroviral Resistance: HIV Prevention at a Cost? *Clinical Infectious Diseases: An Official Publication of the Infectious Diseases Society of America* **53**(12) (Dec. 2011), 1265–1270. doi: 10.1093/cid/cir684 (cit. on p. 98).

597. K. M. Gibas, P. van den Berg, V. E. Powell, and D. S. Krakower: Drug Resistance During HIV Pre-Exposure Prophylaxis. *Drugs* **79**(6) (Apr. 1, 2019), 609–619. doi: 10.1007/s40265-019-01108-x (cit. on p. 98).

598. R. Mourad et al.: A Phylotype-Based Analysis Highlights the Role of Drug-Naive HIV-Positive Individuals in the Transmission of Antiretroviral Resistance in the UK. English. *Aids* **29**(15) (Sept. 24, 2015), 1917–1925. doi: 10.1097/qad.0000000000000768 (cit. on pp. 98, 105).

599. S. Hué et al.: Demonstration of Sustained Drug-Resistant Human Immunodeficiency Virus Type 1 Lineages Circulating among Treatment-Naïve Individuals. en. *Journal of Virology* **83**(6) (Mar. 15, 2009), 2645–2654. doi: 10.1128/jvi.01556-08 (cit. on pp. 98, 105).

600. S. M. Drescher et al.: Treatment-Naive Individuals Are the Major Source of Transmitted HIV-1 Drug Resistance in Men Who Have Sex With Men in the Swiss HIV Cohort Study. *Clinical Infectious Diseases* **58**(2) (Jan. 15, 2014), 285–294. doi: 10.1093/cid/cit694 (cit. on p. 98).

601. R. S. Boerma et al.: High Levels of Pre-Treatment HIV Drug Resistance and Treatment Failure in Nigerian Children. *Journal of the International AIDS Society* **19**(1) (2016), 21140. doi: 10.7448/ias.19.1.21140 (cit. on p. 99).

602. D. S. Clutter, M. R. Jordan, S. Bertagnolio, and R. W. Shafer: HIV-1 Drug Resistance and Resistance Testing. *Infection, Genetics and Evolution* **46** (Dec. 1, 2016), 292–307. doi: 10.1016/j.meegid.2016.08.031 (cit. on p. 99).

603. D. Kühnert et al.: Quantifying the Fitness Cost of HIV-1 Drug Resistance Mutations through Phylodynamics. *PLOS Pathogens* **14**(2) (Feb. 20, 2018), e1006895. doi: 10.1371/journal.ppat.1006895 (cit. on p. 99).

604. T. Mesplède et al.: Viral Fitness Cost Prevents HIV-1 from Evading Dolutegravir Drug Pressure. *Retrovirology* **10**(1) (Feb. 22, 2013), 22. doi: 10.1186/1742-4690-10-22 (cit. on p. 99).

605. H. Castro et al.: Persistence of HIV-1 Transmitted Drug Resistance Mutations. *The Journal of Infectious Diseases* **208**(9) (Nov. 1, 2013), 1459–1463. doi: 10.1093/infdis/jit345 (cit. on p. 99).

606. L. Blassel et al.: Drug Resistance Mutations in HIV: New Bioinformatics Approaches and Challenges. *Current Opinion in Virology* **51** (Dec. 1, 2021), 56–64. doi: 10.1016/j.coviro.2021.09.009 (cit. on p. 99).

607. U. C. S. Committee: The Creation of a Large UK-based Multicentre Cohort of HIV-infected Individuals: The UK Collaborative HIV Cohort (UK CHIC) Study. *HIV Medicine* **5**(2) (2004), 115–124. doi: 10.1111/j.1468-1293.2004.00197.x (cit. on p. 99).

608. L. Abeler-Dörner et al.: PANGEA-HIV 2: Phylogenetics And Networks for Generalised Epidemics in Africa. *Current Opinion in HIV and AIDS* **14**(3) (May 2019), 173–180. doi: 10.1097/coh.0000000000000542 (cit. on p. 99).

609. R. W. Shafer: Rationale and Uses of a Public HIV Drug-Resistance Database. *The Journal of Infectious Diseases* **194** (Supplement_1 Sept. 1, 2006), S51–s58. doi: 10.1086/505356 (cit. on p. 99).

610. C. Kuiken, B. Korber, and R. W. Shafer: HIV Sequence Databases. *AIDS reviews* **5**(1) (2003), 52–61. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2613779/ (cit. on p. 100).

611. A. M. Wensing et al.: 2019 Update of the Drug Resistance Mutations in HIV-1. *Topics in Antiviral Medicine* **27**(3) (Sept. 2019), 111–121. https://pubmed.ncbi.nlm.nih.gov/31634862/ (cit. on p. 100).

612. S. A. Clark, C. Calef, and J. W. Mellors: Mutations in Retroviral Genes Associated with Drug Resistance. *HIV sequence compendium* (2007), 58–158. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.625.1084&rep=rep1&type=pdf (cit. on p. 100).

613. T. F. Liu and R. W. Shafer: Web Resources for HIV Type 1 Genotypic-Resistance Test Interpretation. *Clinical Infectious Diseases* **42**(11) (June 1, 2006), 1608–1618. doi: 10.1086/503914 (cit. on p. 100).

614. V. A. Johnson et al.: Update of the Drug Resistance Mutations in HIV-1: March 2013. *Topics in Antiviral Medicine* **21**(1) (Nov. 28, 2016), 6–7. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6148891/ (cit. on p. 100).

615. C. J. Villabona-Arenas et al.: In-Depth Analysis of HIV-1 Drug Resistance Mutations in HIV-Infected Individuals Failing First-Line Regimens in West and Central Africa. en-US. *Aids* **30**(17) (Nov. 13, 2016), 2577. doi: 10.1097/qad.0000000000001233 (cit. on pp. 100, 106–110, 112, 115).

616. N. S. Shulman, R. J. Bosch, J. W. Mellors, M. A. Albrecht, and D. A. Katzenstein: Genetic Correlates of Efavirenz Hypersusceptibility. *Aids* **18**(13) (Sept. 3, 2004), 1781–1785. doi: 10.1097/00002030-200409030-00006 (cit. on p. 100).

617. M. D. Miller et al.: Genotypic and Phenotypic Predictors of the Magnitude of Response to Tenofovir Disoproxil Fumarate Treatment in Antiretroviral-Experienced Patients. *The Journal of Infectious Diseases* **189**(5) (Mar. 1, 2004), 837–846. doi: 10.1086/381784 (cit. on p. 100).

618. B. W. Brown and K. Russell: Methods Correcting for Multiple Testing: Operating Characteristics. *Statistics in Medicine* **16**(22) (1997), 2511–2528. doi: 10.1002/(sici)1097-0258(19971130)16:22<2511::aid-sim693>3.0.co;2-4 (cit. on p. 100).

619. P. C. Austin, M. M. Mamdani, D. N. Juurlink, and J. E. Hux: Testing Multiple Statistical Hypotheses Resulted in Spurious Associations: A Study of Astrological Signs and Health. *Journal of Clinical Epidemiology* **59**(9) (Sept. 1, 2006), 964–969. doi: 10.1016/j.jclinepi.2006.01.012 (cit. on p. 100).

620. Y. Hochberg and A. C. Tamhane: *Multiple Comparison Procedures*. 1987. doi: 10.1002/9780470316672 (cit. on p. 100).

621. Y. Benjamini and Y. Hochberg: Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society* **57**(1) (1995), 289–300. doi: 10.1111/j.2517-6161.1995.tb02031.x (cit. on p. 100).

622. M. J. Gonzales et al.: Extended Spectrum of HIV-1 Reverse Transcriptase Mutations in Patients Receiving Multiple Nucleoside Analog Inhibitors. *AIDS (London, England)* **17**(6) (Apr. 11, 2003), 791–799. doi: 10.1097/01.aids.0000050860.71999.23 (cit. on p. 100).

623. C. Seoighe et al.: A Model of Directional Selection Applied to the Evolution of Drug Resistance in HIV-1. *Molecular Biology and Evolution* **24**(4) (Apr. 1, 2007), 1025–1031. doi: 10.1093/molbev/msm021 (cit. on p. 100).

624. P. C. Sham and S. M. Purcell: Statistical Power and Significance Testing in Large-Scale Genetic Studies. en. *Nature Reviews Genetics* **15**(5) (May 2014), 335–346. doi: 10.1038/nrg3706 (cit. on pp. 100, 106).

625. S. Alizon et al.: Phylogenetic Approach Reveals That Virus Genotype Largely Determines HIV Set-Point Viral Load. *PLOS Pathogens* **6**(9) (Sept. 30, 2010), e1001123. doi: 10.1371/journal.ppat.1001123 (cit. on p. 100).

626. W. F. Flynn et al.: Deep Sequencing of Protease Inhibitor Resistant HIV Patient Isolates Reveals Patterns of Correlated Mutations in Gag and Protease. *PLOS Computational Biology* **11**(4) (Apr. 20, 2015), e1004249. doi: 10.1371/journal.pcbi.1004249 (cit. on p. 100).

627. C. J. Petropoulos et al.: A Novel Phenotypic Drug Susceptibility Assay for Human Immunodeficiency Virus Type 1. *Antimicrobial Agents and Chemotherapy* **44**(4) (Apr. 2000), 920–928. doi: 10.1128/aac.44.4.920-928.2000 (cit. on p. 100).

628. K. Hertogs et al.: A Rapid Method for Simultaneous Detection of Phenotypic Resistance to Inhibitors of Protease and Reverse Transcriptase in Recombinant Human Immunodeficiency Virus Type 1 Isolates from Patients Treated with Antiretroviral Drugs. *Antimicrobial Agents and Chemotherapy* **42**(2) (Feb. 1998), 269–276. doi: 10.1128/aac.42.2.269 (cit. on p. 100).

629. G. Heilek-Snyder and P. Bean: Role of HIV Phenotypic Assays in the Management of HIV Infection. *American Clinical Laboratory* **21**(1) (2002 Jan-Feb), 40–43. https://europepmc.org/article/med/11975451 (cit. on p. 100).

630. G. J. Moyle et al.: Epidemiology and Predictive Factors for Chemokine Receptor Use in HIV-1 Infection. *The Journal of Infectious Diseases* **191**(6) (Mar. 15, 2005), 866–872. doi: 10.1086/428096 (cit. on p. 100).

631. M. Gartland et al.: Susceptibility of Global HIV-1 Clinical Isolates to Fostemsavir Using the PhenoSense® Entry Assay. *Journal of Antimicrobial Chemotherapy* **76**(3) (Mar. 1, 2021), 648–652. doi: 10.1093/jac/dkaa474 (cit. on p. 100).

632. B. Masquelier et al.: Genotypic and Phenotypic Resistance Patterns of Human Immunodeficiency Virus Type 1 Variants with Insertions or Deletions in the Reverse Transcriptase (RT): Multicenter Study of Patients Treated with RT Inhibitors. *Antimicrobial Agents and Chemotherapy* **45**(6) (June 2001), 1836–1842. doi: 10.1128/aac.45.6.1836-1842.2001 (cit. on p. 100).

633. B. A. Larder and S. D. Kemp: Multiple Mutations in HIV-1 Reverse Transcriptase Confer High-Level Resistance to Zidovudine (AZT). *Science* **246**(4934) (Dec. 1, 1989), 1155–1158. doi: 10.1126/science.2479983 (cit. on p. 100).

634. K. de Vreese et al.: Resistance of Human Immunodeficiency Virus Type 1 Reverse Transcriptase to TIBO Derivatives Induced by Site-Directed Mutagenesis. *Virology* **188**(2) (June 1, 1992), 900–904. doi: 10.1016/0042-6822(92)90550-9 (cit. on p. 100).

635. L. Tambuyzer, S. Nijs, B. Daems, G. Picchio, and J. Vingerhoets: Effect of Mutations at Position E138 in HIV-1 Reverse Transcriptase on Phenotypic Susceptibility and Virologic Response to Etravirine. *JAIDS Journal of Acquired Immune Deficiency Syndromes* **58**(1) (Sept. 1, 2011), 18–22. doi: 10.1097/QAI.0b013e3182237f74 (cit. on p. 100).

636. D. A. Katzenstein et al.: Phenotypic Susceptibility and Virological Outcome in Nucleoside-Experienced Patients Receiving Three or Four Antiretroviral Drugs. *Aids* **17**(6) (Apr. 11, 2003), 821–830. doi: 10.1097/00002030-200304110-00007 (cit. on p. 100).

637. L. Blassel et al.: Using Machine Learning and Big Data to Explore the Drug Resistance Landscape in HIV. *PLOS Computational Biology* **17**(8) (Aug. 26, 2021), e1008873. doi: 10.1371/journal.pcbi.1008873 (cit. on p. 101).

638. O. Sheik Amamuddy, N. T. Bishop, and Ö. Tastan Bishop: Improving Fold Resistance Prediction of HIV-1 against Protease and Reverse Transcriptase Inhibitors Using Artificial Neural Networks. *BMC Bioinformatics* **18**(1) (Aug. 15, 2017), 369. doi: 10.1186/s12859-017-1782-x (cit. on pp. 101, 106).

639. N. Beerenwinkel et al.: Geno2pheno: Interpreting Genotypic HIV Drug Resistance Tests. *IEEE Intelligent Systems* **16**(6) (Nov. 2001), 35–41. doi: 10.1109/5254.972080 (cit. on pp. 101, 107).

640. M. Riemenschneider, T. Hummel, and D. Heider: SHIVA - a Web Application for Drug Resistance and Tropism Testing in HIV. *BMC Bioinformatics* **17**(1) (Aug. 22, 2016), 314. doi: 10.1186/s12859-016-1179-2 (cit. on p. 101).

641. N. Beerenwinkel et al.: Diversity and Complexity of HIV-1 Drug Resistance: A Bioinformatics Approach to Predicting Phenotype from Genotype. *Proceedings of the National Academy of Sciences* **99**(12) (June 11, 2002), 8271–8276. doi: 10.1073/pnas.112177799 (cit. on p. 101).

642. D. Heider, R. Senge, W. Cheng, and E. Hüllermeier: Multilabel Classification for Exploiting Cross-Resistance Information in HIV-1 Drug Resistance Prediction. *Bioinformatics* **29**(16) (Aug. 15, 2013), 1946–1952. doi: 10.1093/bioinformatics/btt331 (cit. on pp. 101, 107).

643. A. C. Lepri et al.: Resistance Profiles in Patients with Viral Rebound on Potent Antiretroviral Therapy. en. *The Journal of Infectious Diseases* **181**(3) (Mar. 1, 2000), 1143–1147. doi: 10.1086/315301 (cit. on p. 105).

644. C. Verhofstede et al.: Detection of Drug Resistance Mutations as a Predictor of Subsequent Virological Failure in Patients with HIV-1 Viral Rebounds of Less than 1,000 RNA Copies/Ml. en. *Journal of Medical Virology* **79**(9) (2007), 1254–1260. doi: 10.1002/jmv.20950 (cit. on p. 105).

645. A. Zhukova, T. Cutino-Moguel, O. Gascuel, and D. Pillay: The Role of Phylogenetics as a Tool to Predict the Spread of Resistance. en. *The Journal of Infectious Diseases* **216**(suppl_9) (suppl_9 Dec. 1, 2017), S820–s823. doi: 10.1093/infdis/jix411 (cit. on p. 105).

646. D. E. Bennett et al.: Drug Resistance Mutations for Surveillance of Transmitted HIV-1 Drug-Resistance: 2009 Update. en. *Plos One* **4**(3) (Mar. 6, 2009), e4724. doi: 10.1371/journal.pone.0004724 (cit. on p. 105).

647. J. Hammond, C. Calef, B. Larder, R. Schinazi, and J. W. Mellors: Mutations in Retroviral Genes Associated with Drug Resistance. en. *Human retroviruses and AIDS* (Dec. 1998), 11136–11179. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.599.5170&rep=rep1&type=pdf (cit. on p. 105).

648. A. M. Wensing et al.: 2017 Update of the Drug Resistance Mutations in HIV-1., 2017 Update of the Drug Resistance Mutations in HIV-1. eng. *Topics in antiviral medicine, Topics in Antiviral Medicine* **24, 24**(4, 4) (4, 4 Dec. 2016), 132, 132–133. https://pubmed.ncbi.nlm.nih.gov/28208121/ (cit. on p. 105).

649. S. Dudoit and M. J. van der Laan: *Multiple Testing Procedures with Applications to Genomics*. en. Dec. 18, 2007. doi: 10.1007/978-0-387-49317-6 (cit. on p. 106).

650. W. P. Maddison and R. G. FitzJohn: The Unsolved Challenge to Phylogenetic Correlation Tests for Categorical Characters. en. *Systematic Biology* **64**(1) (Jan. 1, 2015), 127–136. doi: 10.1093/sysbio/syu070 (cit. on p. 106).

651. T. Lengauer and T. Sing: Bioinformatics-Assisted Anti-HIV Therapy. en. *Nature Reviews Microbiology* **4**(10) (10 Oct. 2006), 790–797. doi: 10.1038/nrmicro1477 (cit. on p. 106).

652. J. Zhang, S.-Y. Rhee, J. Taylor, and R. W. Shafer: Comparison of the Precision and Sensitivity of the Antivirogram and PhenoSense HIV Drug Susceptibility Assays. en-US. *JAIDS Journal of Acquired Immune Deficiency Syndromes* **38**(4) (Apr. 1, 2005), 439–444. doi: 10.1097/01.qai.0000147526.64863.53 (cit. on p. 106).

653. N. Beerenwinkel et al.: Geno2pheno: Estimating Phenotypic Drug Resistance from HIV-1 Genotypes. *Nucleic Acids Research* **31**(13) (July 1, 2003), 3850–3855. doi: 10.1093/nar/gkg575 (cit. on p. 106).

654. C. Shen, X. Yu, R. W. Harrison, and I. T. Weber: Automated Prediction of HIV Drug Resistance from Genotype Data. *BMC Bioinformatics* **17**(8) (Aug. 31, 2016), 278. doi: 10.1186/s12859-016-1114-6 (cit. on p. 106).

655. X. Yu, I. T. Weber, and R. W. Harrison: Prediction of HIV Drug Resistance from Genotype with Encoded Three-Dimensional Protein Structure. *BMC Genomics* **15**(5) (July 14, 2014), S1. doi: 10.1186/1471-2164-15-s5-s1 (cit. on p. 106).

656. S. T. Araya and S. Hazelhurst: Support Vector Machine Prediction of HIV-1 Drug Resistance Using the Viral Nucleotide Patterns. *Transactions of the Royal Society of South Africa* **64**(1) (Jan. 1, 2009), 62–72. doi: 10.1080/00359190909519238 (cit. on p. 107).

657. M. Riemenschneider, R. Senge, U. Neumann, E. Hüllermeier, and D. Heider: Exploiting HIV-1 Protease and Reverse Transcriptase Cross-Resistance Information for Improved Drug Resistance Prediction by Means of Multi-Label Classification. *BioData Mining* **9**(1) (Feb. 2016), 10. doi: 10.1186/s13040-016-0089-1 (cit. on p. 107).

658. S. Dřaghici and R. B. Potter: Predicting HIV Drug Resistance with Neural Networks. *Bioinformatics* **19**(1) (Jan. 1, 2003), 98–107. doi: 10.1093/bioinformatics/19.1.98 (cit. on p. 107).

659. A. C. Mooney et al.: Beyond Social Desirability Bias: Investigating Inconsistencies in Self-Reported HIV Testing and Treatment Behaviors Among HIV-Positive Adults in North West Province, South Africa. en. *AIDS and Behavior* **22**(7) (July 1, 2018), 2368–2379. doi: 10.1007/s10461-018-2155-9 (cit. on p. 109).

660. G. W. Brier: Verification of Forecasts Expressed in Terms of Probability. en. *Monthly Weather Review* **78**(1) (Jan. 1, 1950), 1–3. doi: 10.1175/1520-0493(1950)078<0001:vofeit>2.0.co;2 (cit. on pp. 110, 113).

661. O. Gascuel et al.: Twelve Numerical, Symbolic and Hybrid Supervised Classification Methods. *International Journal of Pattern Recognition and Artificial Intelligence* **12**(05) (Aug. 1, 1998), 517–571. doi: 10.1142/s0218001498000336 (cit. on pp. 110, 111).

662. J. J. Goeman and A. Solari: Multiple Hypothesis Testing in Genomics. en. *Statistics in Medicine* **33**(11) (2014), 1946–1978. doi: 10.1002/sim.6082 (cit. on p. 110).

663. J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger: Tackling the Poor Assumptions of Naive Bayes Text Classifiers. en. *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, 616–623. https://www.aaai.org/Papers/ICML/2003/ICML03-081.pdf (cit. on p. 111).

664. D. Alvarez Melis and T. Jaakkola: Towards Robust Interpretability with Self-Explaining Neural Networks. *Advances in Neural Information Processing Systems 31*. 2018, 7775–7784. http://papers.nips.cc/paper/8003-towards-robust-interpretability-with-self-explaining-neural-networks.pdf (cit. on p. 111).

665. Q. Zhang, Y. N. Wu, and S.-C. Zhu: Interpretable Convolutional Neural Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, 8827–8836. doi: 10.1109/CVPR.2018.00920 (cit. on p. 111).

666. Schrödinger, LLC: The PyMOL Molecular Graphics System, Version 1.8. Nov. 2015 (cit. on p. 118).

667. S.-Y. Rhee, T. F. Liu, S. P. Holmes, and R. W. Shafer: HIV-1 Subtype B Protease and Reverse Transcriptase Amino Acid Covariation. en. *PLOS Computational Biology* **3**(5) (May 11, 2007), e87. doi: 10.1371/journal.pcbi.0030087 (cit. on p. 123).

668. A. De Luca et al.: Improved Interpretation of Genotypic Changes in the HIV-1 Reverse Transcriptase Coding Region That Determine the Virological Response to Didanosine. en. *The Journal of Infectious Diseases* **196**(11) (Dec. 1, 2007), 1645–1653. doi: 10.1086/522231 (cit. on p. 124).

669. A.-G. Marcelin et al.: Impact of HIV-1 Reverse Transcriptase Polymorphism at Codons 211 and 228 on Virological Response to Didanosine. en. *Antiviral Therapy* (2006), 8. doi: 10.1177/135965350601100609 (cit. on p. 124).

670. A. J. L. Brown et al.: Reduced Susceptibility of Human Immunodeficiency Virus Type 1 (HIV-1) from Patients with Primary HIV Infection to Nonnucleoside Reverse Transcriptase Inhibitors Is Associated with Variation at Novel Amino Acid Sites. *Journal of Virology* **74**(22) (Nov. 2000), 10269–10273. doi: 10.1128/jvi.74.22.10269-10273.2000 (cit. on p. 124).

671. S. A. Clark, N. S. Shulman, R. J. Bosch, and J. W. Mellors: Reverse Transcriptase Mutations 118I, 208Y, and 215Y Cause HIV-1 Hypersusceptibility to Non-Nucleoside Reverse Transcriptase Inhibitors. en-US. *Aids* **20**(7) (Apr. 24, 2006), 981–984. doi: 10.1097/01.aids.0000222069.14878.44 (cit. on p. 124).

672. G. Nebbia, C. A. Sabin, D. T. Dunn, and A. M. Geretti: Emergence of the H208Y Mutation in the Reverse Transcriptase (RT) of HIV-1 in Association with Nucleoside RT Inhibitor Therapy. en. *Journal of Antimicrobial Chemotherapy* **59**(5) (May 1, 2007), 1013–1016. doi: 10.1093/jac/dkm067 (cit. on p. 124).

673. A. Saracino et al.: Impact of Unreported HIV-1 Reverse Transcriptase Mutations on Phenotypic Resistance to Nucleoside and Non-Nucleoside Inhibitors. en. *Journal of Medical Virology* **78**(1) (2006), 9–17. doi: 10.1002/jmv.20500 (cit. on p. 124).

674. T. T. Wu, Y. F. Chen, T. Hastie, E. Sobel, and K. Lange: Genome-Wide Association Analysis by Lasso Penalized Logistic Regression. en. *Bioinformatics* **25**(6) (Mar. 15, 2009), 714–721. doi: 10.1093/bioinformatics/btp041 (cit. on p. 124).

675. F. Rosenblatt: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review* **65** (1958), 386–408. doi: 10.1037/h0042519 (cit. on p. 127).

676. D. E. Rumelhart, G. E. Hinton, and R. J. Williams: Learning Representations by Back-Propagating Errors. *Nature* **323**(6088) (6088 Oct. 1986), 533–536. doi: 10.1038/323533a0 (cit. on p. 128).

677. F. Murtagh: Multilayer Perceptrons for Classification and Regression. *Neurocomputing* **2**(5) (July 1, 1991), 183–197. doi: 10.1016/0925-2312(91)90023-5 (cit. on p. 128).

678. G. Cybenko: Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems* **2**(4) (Dec. 1, 1989), 303–314. doi: 10.1007/bf02551274 (cit. on p. 128).

679. K. Hornik, M. Stinchcombe, and H. White: Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks* **2**(5) (Jan. 1, 1989), 359–366. doi: 10.1016/0893-6080(89)90020-8 (cit. on p. 128).

680. K. Hornik: Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks* **4**(2) (Jan. 1, 1991), 251–257. doi: 10.1016/0893-6080(91)90009-t (cit. on p. 128).

681. Y. LeCun et al.: Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation* **1**(4) (Dec. 1989), 541–551. doi: 10.1162/neco.1989.1.4.541 (cit. on p. 128).

682. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner: Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* **86**(11) (Nov. 1998), 2278–2324. doi: 10.1109/5.726791 (cit. on p. 128).

683. J. Voznica et al.: Deep Learning from Phylogenies to Uncover the Epidemiological Dynamics of Outbreaks. *Nature Communications* **13**(1) (1 July 6, 2022), 3896. doi: 10.1038/s41467-022-31511-0 (cit. on p. 128).

684. A. Krizhevsky, I. Sutskever, and G. E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM* **60**(6) (May 24, 2017), 84–90. doi: 10.1145/3065386 (cit. on p. 130).

685. K. He, X. Zhang, S. Ren, and J. Sun: Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, 770–778. https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html (cit. on p. 130).

686. D. Bahdanau, K. Cho, and Y. Bengio: *Neural Machine Translation by Jointly Learning to Align and Translate*. May 19, 2016. doi: 10.48550/arXiv.1409.0473 (cit. on p. 130).

687. A. Vaswani et al.: Attention Is All You Need. *Advances in Neural Information Processing Systems*. **30**. 2017. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (cit. on pp. 130, 131).

688. *How Many Words Are There in English? | Merriam-Webster*. Merriam-Webster. https://www.merriam-webster.com/help/faq-how-many-english-words (cit. on p. 130).

689. T. Mikolov, K. Chen, G. Corrado, and J. Dean: *Efficient Estimation of Word Representations in Vector Space*. Sept. 6, 2013. doi: 10.48550/arXiv.1301.3781 (cit. on p. 130).

690. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean: Distributed Representations of Words and Phrases and Their Compositionality. *Advances in Neural Information Processing Systems*. **26**. 2013. https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html (cit. on p. 130).

691. Y. Goldberg and O. Levy: *Word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method*. Feb. 15, 2014. doi: 10.48550/arXiv.1402.3722 (cit. on p. 130).

692. P. Ng: *Dna2vec: Consistent Vector Representations of Variable-Length k-Mers*. Jan. 23, 2017. doi: 10.48550/arXiv.1701.06279 (cit. on p. 131).

693. Y. Liang et al.: Hyb4mC: A Hybrid DNA2vec-based Model for DNA N4-methylcytosine Sites Prediction. *BMC Bioinformatics* **23**(1) (June 29, 2022), 258. doi: 10.1186/s12859-022-04789-6 (cit. on p. 131).

694. D. Kimothi, A. Soni, P. Biyani, and J. M. Hogan: *Distributed Representations for Biological Sequence Analysis*. Sept. 12, 2016. doi: 10.48550/arXiv.1608.05949 (cit. on p. 131).

695. E. Asgari and M. R. K. Mofrad: Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLoS ONE* **10**(11) (Nov. 10, 2015), e0141287. doi: 10.1371/journal.pone.0141287 (cit. on p. 131).

696. D. Kimothi, A. Shukla, P. Biyani, S. Anand, and J. M. Hogan: Metric Learning on Biological Sequence Embeddings. *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. July 2017, 1–5. doi: 10.1109/spawc.2017.8227769 (cit. on p. 131).

697. B. Song et al.: Pretraining Model for Biological Sequence Data. *Briefings in Functional Genomics* **20**(3) (May 28, 2021), 181–195. doi: 10.1093/bfgp/elab025 (cit. on p. 131).

698. H. Wang, H. Wu, Z. He, L. Huang, and K. Ward Church: Progress in Machine Translation. *Engineering* (July 14, 2021). doi: 10.1016/j.eng.2021.03.023 (cit. on p. 131).

699. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. May 24, 2019. doi: 10.48550/arXiv.1810.04805 (cit. on p. 131).

700. T. Brown et al.: Language Models Are Few-Shot Learners. *Advances in Neural Information Processing Systems*. **33**. 2020, 1877–1901. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (cit. on p. 131).

701. A. Madani et al.: ProGen: Language Modeling for Protein Generation. *bioRxiv* (Mar. 8, 2020). doi: 10.1101/2020.03.07.982272 (cit. on p. 132).

702. Erik Nijkamp, Jeffrey A. Ruffolo, Eli N. Weinstein, Nikhil Naik, and Ali Madani: ProGen2: Exploring the Boundaries of Protein Language Models. *ArXiv*(arXiv:2206.13517) (2022). doi: 10.48550/arxiv.2206.13517 (cit. on p. 132).

703. T. Bepler and B. Berger: Learning the Protein Language: Evolution, Structure, and Function. *Cell systems* **12**(6) (June 16, 2021), 654–669.e3. doi: 10.1016/j.cels.2021.05.017 (cit. on p. 132).

704. R. Rao, J. Meier, T. Sercu, S. Ovchinnikov, and A. Rives: *Transformer Protein Language Models Are Unsupervised Structure Learners*. Dec. 15, 2020. doi: 10.1101/2020.12.15.422761 (cit. on p. 132).

705. A. Rives et al.: Biological Structure and Function Emerge from Scaling Unsupervised Learning to 250 Million Protein Sequences. *bioRxiv* **118**(15) (Apr. 29, 2019), 622803. doi: 10.1101/622803 (cit. on p. 132).

706. N. Bhattacharya et al.: *Single Layers of Attention Suffice to Predict Protein Contacts*. Dec. 22, 2020. doi: 10.1101/2020.12.21.423882 (cit. on p. 132).

707. M. Hu et al.: *Exploring Evolution-Based & -Free Protein Language Models as Protein Function Predictors*. June 13, 2022. doi: 10.48550/arXiv.2206.06583 (cit. on p. 132).

708. J. Meier et al.: Language Models Enable Zero-Shot Prediction of the Effects of Mutations on Protein Function. *bioRxiv* **34** (Dec. 6, 2021), 29287–29303. doi: 10.1101/2021.07.09.450648 (cit. on p. 132).

709. B. Hie, Kevin K Yang, and S. K. Kim: Evolutionary Velocity with Protein Language Models Predicts Evolutionary Dynamics of Diverse Proteins. *Cell systems* **13**(4) (Feb. 1, 2022), 274–285.e6. doi: 10.1016/j.cels.2022.01.003 (cit. on p. 132).

710. Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri: DNABERT: Pre-Trained Bidirectional Encoder Representations from Transformers Model for DNA-language in Genome. *Bioinformatics* **37**(15) (Aug. 1, 2021), 2112–2120. doi: 10.1093/bioinformatics/btab083 (cit. on p. 132).

711. G. Benegas, S. S. Batra, and Y. S. Song: *DNA Language Models Are Powerful Zero-Shot Predictors of Non-Coding Variant Effects*. Aug. 23, 2022. doi: 10.1101/2022.08.22.504706 (cit. on p. 132).

712. T. Cai et al.: *Genome-Wide Prediction of Small Molecule Binding to Remote Orphan Proteins Using Distilled Sequence Alignment Embedding*. Aug. 5, 2020. doi: 10.1101/2020.08.04.236729 (cit. on p. 132).

713. R. Rao et al.: MSA Transformer. *bioRxiv* (2021). doi: 10.1101/2021.02.12.430858 (cit. on p. 132).

714. T. Sercu et al.: *Neural Potts Model*. Apr. 11, 2021. doi: 10.1101/2021.04.08.439084 (cit. on p. 132).

715. P. Sturmfels, J. Vig, A. Madani, and N. F. Rajani: *Profile Prediction: An Alignment-Based Pre-Training Task for Protein Sequence Models*. Nov. 30, 2020. doi: 10.48550/arXiv.2012.00195 (cit. on p. 132).

716. G. Baid et al.: DeepConsensus Improves the Accuracy of Sequences with a Gap-Aware Sequence Transformer. *Nature Biotechnology* (Sept. 1, 2022), 1–7. doi: 10.1038/s41587-022-01435-7 (cit. on p. 132).

717. A. Ourmazd, K. Moffat, and E. E. Lattman: Structural Biology Is Solved — Now What? *Nature Methods* **19**(1) (1 Jan. 2022), 24–26. doi: 10.1038/s41592-021-01357-3 (cit. on p. 132).

718. J. Vig et al.: *BERTology Meets Biology: Interpreting Attention in Protein Language Models*. Mar. 28, 2021. doi: 10.48550/arXiv.2006.15222 (cit. on p. 132).

719. M. Gao and J. Skolnick: A Novel Sequence Alignment Algorithm Based on Deep Learning of the Protein Folding Code. *Bioinformatics* **37**(4) (Feb. 15, 2021), 490–496. doi: 10.1093/bioinformatics/btaa810 (cit. on p. 133).

720. J. T. Morton et al.: *Protein Structural Alignments From Sequence*. Nov. 4, 2020. doi: 10.1101/2020.11.03.365932 (cit. on p. 133).

721. H. Berman, K. Henrick, H. Nakamura, and J. L. Markley: The Worldwide Protein Data Bank (wwPDB): Ensuring a Single, Uniform Archive of PDB Data. *Nucleic Acids Research* **35** (suppl_1 Jan. 1, 2007), D301–d303. doi: 10.1093/nar/gkl971 (cit. on p. 133).

722. Y. Guo, J. Wu, H. Ma, S. Wang, and J. Huang: Comprehensive Study on Enhancing Low-Quality Position-Specific Scoring Matrix with Deep Learning for Accurate Protein Structure Property Prediction: Using Bagging Multiple Sequence Alignment Learning. *Journal of Computational Biology* **28**(4) (Apr. 2021), 346–361. doi: 10.1089/cmb.2020.0416 (cit. on p. 133).

723. F. Llinares-López, Q. Berthet, M. Blondel, O. Teboul, and J.-P. Vert: *Deep Embedding and Alignment of Protein Sequences*. July 1, 2022. doi: 10.1101/2021.11.15.468653 (cit. on p. 133).

724. B. E. Suzek et al.: UniRef Clusters: A Comprehensive and Scalable Alternative for Improving Sequence Similarity Searches. *Bioinformatics* **31**(6) (Mar. 15, 2015), 926–932. doi: 10.1093/bioinformatics/btu739 (cit. on p. 133).

725. J. Mistry et al.: Pfam: The Protein Families Database in 2021. *Nucleic Acids Research* **49**(D1) (Jan. 8, 2021), D412–d419. doi: 10.1093/nar/gkaa913 (cit. on p. 134).

726. S. Petti et al.: *End-to-End Learning of Multiple Sequence Alignments with Differentiable Smith-Waterman*. Apr. 18, 2022. doi: 10.1101/2021.10.23.465204 (cit. on p. 134).

727. E. Dotan et al.: *Harnessing Machine Translation Methods for Sequence Alignment*. July 23, 2022. doi: 10.1101/2022.07.22.501063 (cit. on p. 134).

728. S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma: *Linformer: Self-Attention with Linear Complexity*. June 14, 2020. doi: 10.48550/arXiv.2006.04768 (cit. on p. 135).

729. Y. Xiong et al.: Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention. *Proceedings of the AAAI Conference on Artificial Intelligence* **35**(16) (16 May 18, 2021), 14138–14148. doi: 10.1609/aaai.v35i16.17664 (cit. on p. 135).

730. R. Child, S. Gray, A. Radford, and I. Sutskever: *Generating Long Sequences with Sparse Transformers*. Apr. 23, 2019. doi: 10.48550/arXiv.1904.10509 (cit. on p. 135).

731. G. M. Correia, V. Niculae, and A. F. T. Martins: Adaptively Sparse Transformers. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China, Nov. 2019, 2174–2184. doi: 10.18653/v1/D19-1223 (cit. on p. 135).

732. S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin: *Adaptive Attention Span in Transformers*. Aug. 8, 2019. doi: 10.48550/arXiv.1905.07799 (cit. on p. 135).

733. Z. Wu, Z. Liu, J. Lin, Y. Lin, and S. Han: *Lite Transformer with Long-Short Range Attention*. Apr. 24, 2020. doi: 10.48550/arXiv.2004.11886 (cit. on p. 135).

734. N. Kitaev, Ł. Kaiser, and A. Levskaya: *Reformer: The Efficient Transformer*. Feb. 18, 2020. doi: 10.48550/arXiv.2001.04451 (cit. on p. 135).

735. K. Choromanski et al.: *Masked Language Modeling for Proteins via Linearly Scalable Long-Context Transformers*. Sept. 30, 2020. doi: 10.48550/arXiv.2006.03555 (cit. on p. 135).

736. N. Bhattacharya et al.: Interpreting Potts and Transformer Protein Models Through the Lens of Simplified Attention. *Biocomputing 2022*. Sept. 21, 2021, 34–45. doi: 10.1142/9789811250477_0004 (cit. on p. 135).

737. T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis: The Case for Learned Index Structures. *Proceedings of the 2018 International Conference on Management of Data*. New York, NY, USA, May 27, 2018, 489–504. doi: 10.1145/3183713.3196909 (cit. on p. 136).

738. Y. Jung and D. Han: BWA-MEME: BWA-MEM Emulated with a Machine Learning Approach. *Bioinformatics* **38**(9) (May 1, 2022), 2404–2413. doi: 10.1093/bioinformatics/btac137 (cit. on p. 136).

739. M. Kirsche, A. Das, and M. C. Schatz: Sapling: Accelerating Suffix Array Queries with Learned Data Models. *Bioinformatics* **37**(6) (Mar. 15, 2021), 744–749. doi: 10.1093/bioinformatics/btaa911 (cit. on p. 136).

740. D. Ho et al.: *LISA: Learned Indexes for Sequence Analysis*. July 13, 2021. doi: 10.1101/2020.12.22.423964 (cit. on p. 136).

741. M. Hoang, H. Zheng, and C. Kingsford: Differentiable Learning of Sequence-Specific Minimizer Schemes with DeepMinimizer. *Journal of Computational Biology* (Sept. 12, 2022). doi: 10.1089/cmb.2022.0275 (cit. on p. 136).

742. S. Min, B. Lee, and S. Yoon: TargetNet: Functional microRNA Target Prediction with Deep Neural Networks. *Bioinformatics* **38**(3) (Feb. 1, 2022), 671–677. doi: 10.1093/bioinformatics/btab733 (cit. on p. 136).

743. A. V. Bzikadze and P. A. Pevzner: Automated Assembly of Centromeres from Ultra-Long Error-Prone Reads. *Nature Biotechnology* **38**(11) (11 Nov. 2020), 1309–1316. doi: 10.1038/s41587-020-0582-4 (cit. on p. 143).

744. J. Castresana: Selection of Conserved Blocks from Multiple Alignments for Their Use in Phylogenetic Analysis. en. *Molecular Biology and Evolution* **17**(4) (Apr. 1, 2000), 540–552. doi: 10.1093/oxfordjournals.molbev.a026334 (cit. on p. 158).

745. F. Pedregosa et al.: Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research* **12**(Oct) (Oct 2011), 2825–2830. http://www.jmlr.org/papers/v12/pedregosa11a.html (cit. on pp. 158, 159).

746. P. Virtanen et al.: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. en. *Nature Methods* **17**(3) (3 Mar. 2020), 261–272. doi: 10.1038/s41592-019-0686-2 (cit. on p. 158).

747. S. Seabold and J. Perktold: Statsmodels: Econometric and Statistical Modeling with Python. en. *Proceedings of the 9th Python in Science Conference*. Austin, Texas, 2010, 92–96. doi: 10.25080/Majora-92bf1922-011 (cit. on p. 158).

748. N. X. Vinh and J. Epps: A Novel Approach for Automatic Number of Clusters Detection in Microarray Data Based on Consensus Clustering. *2009 Ninth IEEE International Conference on Bioinformatics and BioEngineering*. June 2009, 84–91. doi: 10.1109/bibe.2009.19 (cit. on p. 159).

749. P. Harremoes: Mutual Information of Contingency Tables and Related Inequalities. *2014 IEEE International Symposium on Information Theory*. Honolulu, HI, USA, Feb. 1, 2014, 2474–2478. doi: 10.1109/isit.2014.6875279 (cit. on p. 159).

Abstract

From sequences to knowledge, improving and learning from sequence alignments.

In this thesis we study two important problems in computational biology, one pertaining to primary analysis of sequencing data, and the second pertaining to secondary analysis of sequences to obtain biological insights using machine-learning. Sequence alignment is one of the most powerful and important tools in the field of computational biology. Read alignment is often the first step in many analyses like structural variant detection, genome assembly or variant calling. Long read sequencing technologies have improved the quality of results across all these analyses. They remain, however, plagued by sequencing errors and pose algorithmic challenges to alignment. A prevalent technique to reduce the detrimental effects of these errors is homopolymer compression, which targets the most common type of long-read sequencing error. We present a more general framework than homopolymer compression, which we call mapping-friendly sequence reductions (MSR). We then show that some of these MSRs improve the accuracy of read alignments across whole human, *drosophila* and *E. coli* genomes. Improvements in sequence alignment methods are crucial for downstream analyses. For instance, multiple sequence alignments are indispensable when studying resistance in viruses. With the ever growing quantity of annotated, high-quality multiple sequence alignments it has become possible and useful to study drug resistance in viruses with machine learning methods. We used a very large multiple sequence alignment of British HIV sequences to train multiple classifiers to discriminate between treatment-naive and treatment-experienced sequences. By studying important classifier features we identified resistance-associated mutations. We then removed known drug resistance associated signal from the data before training, keeping classifying power, and identified 6 novel resistance associated mutations. Further study indicated that these were most likely accessory in nature and linked to known resistance mutations.

Keywords: Alignment, Genomics, Machine Learning, Biological sequences

Résumé

Des séquences au savoir, améliorer et apprendre des alignements de séquences.

Dans cette thèse nous étudierons deux problèmes importants en bioinformatique, le premier concernant l'analyse primaire de données de séquencage, et le second concernant l'analyse secondaire de séquence par apprentissage automatique en vue d'obtenir des connaissances biologiques. L'alignement de séquences est l'un des outils les plus puissants et les plus importants dans le domaine de la biologie computationnelle. L'alignement de lectures de séquencage est souvent la première étape de nombreuses analyses telles que la détection de variations de structure, ou l'assemblage de génomes. Les technologies de séquençage à longue lectures ont amélioré la qualité des résultats pour toutes ces analyses. Elles sont, cependant, riches en erreurs de séquençage et posent des problèms algorithmiques à l'alignement. Une technique répandue pour réduire les effets néfastes de ces erreurs est la compression d'homopolymères. Cette technique cible le type d'erreur de séquençage à longue lectures le plus fréquent. Nous présentons une technique plus générale que la compression d'homopolymères, que nous appelons les "mapping-friendly sequence reductions" (MSR). Nous montrons ensuite que certaines de ces MSRs améliorent la précision des alignements de lecture sur des génomes entiers d'humain, de *drosophile* et d'*E. coli*. L'amélioration des méthodes d'alignment de séquences est cruciale pour les analyses en aval. Par exemple, les alignements de séquences multiples sont indispensables pour étudier la pharmaco-résistance des virus. Grâce à la quantité toujours croissante d'alignements de séquences multiples annotés et de haute qualité, il est aujourd'hui devenu possible et utile d'étudier la résistance des virus à l'aide de méthodes d'apprentissage automatique. Nous avons utilisé un très grand alignement de séquences multiples de séquences de VIH britanniques et entraîné plusieurs classificateurs pour distinguer les séquences non-traitées des séquences traitées. En étudiant les variables importantes aux classificateurs, nous identifions des mutations associées à la résistance. Nous avons ensuite supprimé des données, avant l'entraînement, le signal de pharmaco-résistance connu. Nous conservons le pouvoir discriminant des classificateurs, et avons identifié 6 nouvelles mutations associées à la résistance. Une étude plus approfondie a montré que celles-ci étaient très probablement accessoires et liées à des mutations de résistance connues.

Mots clés: Alignement, Génomique, Machine Learning, Séquence biologiques