Algorithmic approaches to biological sequence analysis generate new tools for the study of genome structure and function.

Ing. Matej LEXA, PhD

ABSTRACT

This work provides background information and commentary to 6 research papers I co-authored. It describes my entry into and research in bioinformatics from the year 2001 until today. During this time period I explored a number of avenues in biological sequence analysis, proposed and designed new algorithms or redressed well-known algorithms to be applied in novel applications to biological data. The problems that interested me most were:

- How to search and analyze biological sequences in minimum time and space? The answer lied mostly in specialized k-mer data structures and algorithms combined with implementations relying on pointer arithmetics and bit operations.
- Can we identify and evaluate specific DNA subsequences capable of forming interesting structures at the molecular level? I found two novel approaches based on effective evaluation of nucleotide triplets and quadruplets and their mapping to known structures.
- What kind of genomic DNA sequence analysis is necessary to aid biologists in analyzing genome evolution and 3-D structure? It turned out that existing approaches often ignored repetitive regions in DNA, so I concentrated on those and designed computational tools for their annotation.

All the algorithmic ideas were swiftly implemented either in research software prototypes or in more advanced software packages made available to the research community via appropriate channels, such as repository-deposited source code accompanied by publications, downloadable GNU Linux packages made with automake or R/Bioconductor packages. They include the following algorithms/software:

1. **Virtual PCR** - a dynamic simulation model with a biological sequence mining component to predict the results of polymerase chain reactions; written in Perl and C

2. **PRIMEX –** C++/Perl implementation of a k-mer hashing-based short sequence alignment algorithm; an offline string search approach to genome analysis
3. **Triplex –** a new dynamic programming algorithm capable of predicting triplex DNA formation from sequence; implemented as an R/Bioconductor package
4. **PQSfinder –** an algorithm for exhaustive but efficient search in nucleotide sequence space to identify DNA sequence capable of forming G-guadruplex structures implemented as an R/Bioconductor package; written in C++/R
5. **TE-greedy-nester –** a greedy algorithm to identify nested transposable elements in genomic DNA sequences implemented in Python
6. **HiC-TE –** a Nextflow workflow based a novel sequencing data analysis paradigm combining existing software with components written in bash, perl, R and Python

All six commented papers were published in Bioinformatics, considered one of the top journals in this area, however I also mention and cite a number of other papers I co-authored. They mostly represent the application of the developed approaches to important biological problems.

GLOSSARY

Base - a special chemical group differing between basic building blocks of DNA, commonly designated as A,C,G,T

Basepair (bp) - a term for two bases used when referring to the length of a DNA molecule or sequence; bases are typically organized in one of two pairs, A-T or C-G

(DNA) clone - a fragment of a DNA molecule that is copied and worked on in a laboratory

DNA - the molecule carrying genetic information found in cells (mostly in the nucleus, organized into chromosomes); the information it carries is coded into a sequence (or string) of bases (or basepairs); it is "read" by the cell to make RNA and protein

DNA or genome sequencing - identification of the precise order of bases/basepairs in the DNA molecules of a given sample or organism

Eukaryote - a type of organisms made of cells that have a nucleus; this group encompasses fungi, plants and animals; bacteria, on the other hand belong to a different group called prokaryotes

Genome - a collection of all genes of a given organism; however it is commonly used in a wider sense to mean all the DNA of an organism

G-quadruplex - a special structure that can be formed by DNA or RNA rich in guanines. Tetrads of guanine bases (the Gs) are formed by Hoogsteen bonding as two or more of these get stabilized by potassium ions and can thereafter form a stable structure with poorly understood/described biological functions

Hoogsteen basepair - chemically and structurally a different kind of basepair than the canonical Watson-Crick; this kind of bonding between bases is seen in triplex and quadruplex DNA

K-mer - a short substring of a string of length K

LTR retrotransposon - a special class of transposable elements that spread by a "copy-paste" mechanism via an RNA intermediate and are then reversely (retro-) transcribed back into the genomic DNA

Nucleic acid - a biochemical substance, a polymer (long string) of nucleotides; includes DNA and RNA

Protein - a polymer of amino acids; in cells the information regading what amino acids a protein should be made of is present in the form of genes in DNA molecules forming the genome; there are many different genes in a genome and many different proteins are made during the lifetime of a cell; if

we were to use an analogy where the genome is the “software” that the cell carries along, then proteins would be the “hardware” of the cell

PCR (polymerase chain reaction) - a biochemical reaction used in the laboratory to synthesize a certain fragment of DNA many times over, preserving the sequence of one or a few DNA molecules that are used as a template to be copied; the copying/synthesis is chemically carried out by a DNA polymerase enzyme that has to be added to the reaction together with the building blocks of the new DNA (nucleotides) and short fragments of DNA (primers) from which the synthesis will always start and which delineate the borders of the desired fragment on the template by binding to form a duplex

RNA - a type of nucleic acid that carries information for protein synthesis around the cell as mRNA; a few specialized species of RNA carry out slightly different functions (e.g. t-RNA, rRNA); in contrast to DNA, RNA is typically single-stranded and not strictly helical

(Sequence) assembly - the process of putting together many small sequencing reads (short sequences identified by sequencing (machines)) in order to reconstruct the sequence of the whole genome that was sequenced. Most sequencing technologies are unable to read the sequence of bases of an entire DNA molecule in one reading.

Transposable element – a region of DNA in the genome that has the ability to act autonomously by moving or being copied to a different place in the genome; transposable elements are the most common cause for dispersed repeats – the phenmenon of encountering the same DNA sequence many times in different regions of the genome

Triplex DNA - DNA molecules in which a third strand is attached to the usual helical DNA duplex; intermolecular complexes of this kind are called H-DNA

INTRODUCTION

The last two decades of biological and medical research has been marked by ground-breaking progress enabled by the arrival of new technologies and methods in the area of *molecular biology* and *genomics*. While molecular biology as a discipline has roots in the previous century when the structure of proteins and nucleic acids was discovered, genomics is the younger sister of molecular biology, providing methods and tools to study life at molecular level using highly parallel techniques. Genomic approaches have in turn flooded the scientific arena with unprecedented volumes of raw data, mostly biological sequences and their annotation in time and space. These developments resulted in an ever increasing reliance on computers and algorithmic solutions in biomedicine, ushering in the era of *bioinformatics*.

Twenty years ago it were the physicists with particle accelerators and colliders and astronomers with powerful telescopes who were the hottest candidates for big data accumulation and analysis. Today, genomics, or biology and medicine are quickly becoming one of the most data-intensive scientific disciplines on the planet. Having the data available, however, is only the beginning of a long journey.

In the year 2001, the Human Genome Project yielded the long-awaited fruits of its labor, providing us with roughly 3 billion nucleotide bases, the As, C, Gs and Ts of the first reference human genome. Even with such a modest volume of new data, the task of *converting this data into knowledge* turned out to be a winding road with a lot of dead-ends. For example, a group of people at the forefront of medical research and human genome sequencing in the US, William A. Haseltine and Craig Venter, founded a company called Human Genome Sciences back in 1992 with a vision to build upon the sequencing of the human genome and profit on the ability to use this information to understand the underlying molecular causes behind disease and design medicine tailored to the specific mechanisms discovered in genomic data (Allen 2005, McNamee and Ledley, 2013). After almost ten years of studying the human genome, the company spent all the venture capital money, about 4 billion USD, only to come up with one candidate drug to treat lupus. Even that drug was not approved for everybody in a need of such medicine. The company was taken over by a pharmaceutical whale Glaxo-Smith-Kline before it could go bankrupt.

On the other hand, there are many success stories. They underline the fact that understanding the molecular sequences and structures of life is far from straightforward, but when done with the right tools and mindset, all kinds of basic and practical/actionable knowledge can be derived from these data. To name just a few, one can look at the scientific response to the surprising arrival of the SARS-CoV2 virus. With the ability to isolate and sequence the virus, molecular biologists and other practitioners of science and medicine were able to synthesize an artificial RNA molecule to be used as a vaccine. Later, they were able to do the same with many viral samples, compare the sequences to each other and follow the spread of different variants and lineages. None of this would be possible without necessary genomic and bioinformatic tools already in place, ready to be used at the onset of the pandemic. Another example of an enormous dataset and a resulting biological insight is a recent supercomputing exercise in petabase-scale virus discovery (Edgar et al., 2022). Perhaps the best description of this study is what the authors of the study wrote themselves in the abstract of the above Nature paper:

> *Public databases contain a planetary collection of nucleic acid sequences, but their systematic exploration has been inhibited by a lack of efficient methods for searching this corpus, which (at the time of writing) exceeds 20 petabases and is growing exponentially*[1]*. Here we developed a cloud computing infrastructure, Serratus, to enable ultra-high-throughput sequence alignment at the petabase scale. We searched 5.7 million biologically diverse samples (10.2 petabases) for the hallmark gene RNA-dependent RNA polymerase and identified well over 105 novel RNA viruses, thereby expanding the number of known species by roughly an order of magnitude.*

These examples illustrate that currently there are several ways to convert data into knowledge (Gagneur et al., 2017). One is to increase the output of the analysis of such data to the point that high-performance computing can find interesting patterns (as shown in the above example). If the patterns to look for are not trivial or not even known, machine learning methods can be trained in what will probably be black-box models of life and all its intricacies, nevertheless models that will most likely provide practical solutions to many problems that were difficult to address only a decade ago[2]. Another

1 An account of the controversies surrounding the first RNA sequence of the virus see Campbell (2020) “Exclusive: The Chinese Scientist Who Sequenced the First COVID-19 Genome Speaks Out About the Controversies Surrounding His Work” in Aug 24, 2020 issue of TIME magazine (url: https://time.com/5882918/zhang-yongzhen-interview-china-coronavirus-genome/).

2 AlphaFold protein structure prediction is one of the first better known techniques of this kind (url: https://alphafold.com/).

is to improve our *mechanistic understanding of life* at molecular level to the point that we can understand the function of most of the sequences present in past and current genomes of various species of life. Both approaches require not only more data by volume or type but also better ability to store and analyze it, and ultimately draw conclusions from them. Be it in the form of new knowledge or practical guidelines for the given moment. In what amounts to about one lifetime, we witnessed a gradual expansion of the scope of biological research to the arena of molecular biology, further to genomics, with bioinformaticians as a necessary chain link who can help to make sense of the accumulated data by following some of the paths outlined above.

In my work, first as a biologist, later (since cca 2001) as a bioinformatician, I always strived to place myself at the *intersection of biology and computer science*. First in embracing numerical simulation models to explain phenomena in agriculture, plant nutrition and plant physiology and biochemistry. Later, with the increasing importance of molecular biology, genomics and bioinformatics, I shifted interests slightly, from analog to digital, from simulation models to biological sequence analysis. Symbolically, my first work in this area combined a dynamic mathematical model of DNA fragment binding and synthesis in PCR (polymerase chain reaction; a commonly used technique for detection or synthesis of DNA molecules in the laboratory) with a sequence mining approach that required the adaptation and implementation of a string matching algorithm. This happened between 2001-2003 and it marked a new era in my scientific career where I concentrated on efficient and practical ways of biological sequence analysis and the application of the resulting tools and algorithms to some pressing and interesting biological problems at the time.

My contribution to this field is demonstrated here in the 6 attached publications, each with its own chapter and commentary.

INCLUDED PUBLICATIONS

	Publication	Contribution
1	Lexa et al. 2001 – Virtual PCR	design and implementation of simulation model (100%) data collection (50%) writing (90%)
2	Lexa et al. 2003 – Primex: rapid identification of oligonucleotide matches in whole genomes	design and implementation of string matching algorithm (100%) writing (90%)
3	Lexa et al. 2011 - A dynamic programming algorithm for identification of triplex-forming sequences	idea for the algorithm (80%) data and implementation (20%) writing (80%)
4	Hon et al. 2017 - pqsfinder: an exhaustive and imperfection-tolerant search tool for potential quadruplex-forming sequences in R	idea for the algorithm (40%) data and implementation (10%) writing (80%)
5	Lexa et al. 2020 - TE-greedy-nester: structure-based detection of LTR retrotransposons and their nesting	design of search and classification algorithm (80%) implementation and data processing (30%) writing (80%)
6	Lexa et al. 2021 - HiC-TE: a computational pipeline for Hi-C data analysis to study the role of repeat family interactions in the genome 3D organization	idea for the algorithm (100%) design and implementation of pipeline (50%) data analysis (80%) writing (80%)

Lexa et al (2001). Virtual PCR. Bioinformatics 17(2):192–193.
doi: 10.1093/bioinformatics/17.2.192

Contribution: design and implementation of simulation model (100%), data collection (50%), writing (90%)
WoS citations: **28** *Google Scholar citations*: **64**

Towards the end of the millenium, biologists have already collected a considerable amount of sequence data. The central repository for such data was the NCBI GenBank sequence database (Benson et al., 2000). The data did not come from the massively parallel techniques used today but from many small experiments carried out in laboratories around the world (Bilofsky and Burks, 1988). Consequently, it contained a heterogeneous mix of sequence data from short clones collected and analyzed to find the DNA sequence of a fragment ranging from a single gene or transcript, to medium-sized BAC clones and other sequences. These were collected systematically for many model genome species in the process of genome sequencing, or as a cheaper alternative to assembling the full genome by simply sampling it.

I realized this data could be mined in a way that would support decision making in a number of real-world research situations. In one laboratory technique, called PCR, a fragment of DNA is synthesized by choosing a pair of short sequences upstream and downstream of the sequence of interest and then DNA polymerase enzyme is used to make new DNA molecules with identical sequence as the original fragment (template), starting alternatively from one primer or the other (Figure 1). At that time, it was possible to choose a good primer using software that evaluated several properties of the primer DNA sequences, such as Primer3 (Rozen and Skaletsky, 2000). However, this approach did not look at one important parameter, the uniqueness of the primer binding sequence within the template, especially when amplifying parts of a genome. Often, PCR is used on material that contains the entire genome of a species, or even more than one genome (Jung et al. 1992). Also, often the primers are intentionally made less specific, leading to the so-called multiplex PCR, where the process may result in the synthesis of several different DNA products (Edwards and Gibbs, 1994). PCR is also used in a

diagnostic manner, as a proof of the presence of some primer-binding sequence in the analyzed material (Grondahl et al., 1999).

Compared to Primer3(Rychlik and Roads, 1989) and similar software, I imagined an improved computational approach that would consult the NCBI GenBank database for any primer pair being tested and help the user determine what product to expect from running a PCR with these primers. Such software would have two main components. Given a pair of primers and the name of the analyzed species as input, one component would find all sequences similar enough so that they could bind the primers (this part initially relied on existing BLAST software (Altschul et al., 1990)), while the second component would then run a dynamic simulation model of the PCR reaction and predict the synthesized (also known as amplified) DNA products. If the primers were not specific enough, or more than two were used as input, multiple products were predicted, showing the ability to simulate DNA products even for multiplex PCR. The software was written in C and Perl. The model was written as a system of differential equations that were solved using the Euler method (Atkinson, 1989).

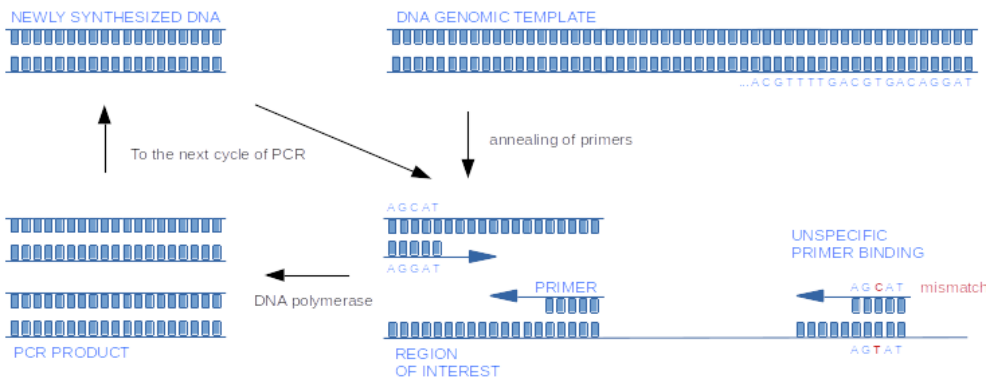

Figure 1. The role of primers in the PCR reaction and the need to detect unspecific primer binding.

At the time of publication there was no such software available to researchers, making both, the idea and the software quite popular. I succeeded receiving a 2-year research fellowship from the University of Padua (Padova) to work with professor Giorgio Valle on further developing and improving this computational approach. While in Padova, I improved the simulation model to use the latest methods in melting temperature prediction for DNA duplexes (a primer bound to its template in PCR forms a primer-template duplex only at certain temperatures) as proposed by SantaLucia Jr. (1998) and further

fine-tuned the simulation process. The paper describing the software published in Bioinformatics is my 4th most cited paper on Google Scholar and the 8th most cited on Web of Science. Ideas from the paper were later used in other primer design software, such as MFEprimer (Qu et al., 2009, Wang et al., 2019), Genomemasker (Anderson et al., 2005), Puns (Boutros and Okey, 2004) and FastPCR (Kalendar et al., 2017). Several research groups used the software to evaluate large sets of primers as a filtration step, to select only primers that showed promising results in the simulation. For example, Laganiere et al. (2005) write: "*Primer pairs were designed by using the Primer3 algorithm, and the specificity was tested in silico by using a virtual PCR algorithm*".

While still at Padova University, I designed and implemented a new module of Virtual PCR later called PRIMEX (see next chapter). It replaced BLAST sequence matching to NCBI databases with full genome searches, using a new algorithm based on short fixed word (k-mer) counting and hashing, thus speeding up the prediction from tens of minutes to minutes or even seconds on a single typical contemporary computer. The speedup was much higher for large genomes, which at the time were beginning to be deposited in the NCBI database (Arabidopsis, human and mouse genomes, for example). The PRIMEX string matching software is fully described in the next chapter.

Lexa et al. 2003 – Primex: rapid identification of oligonucleotide matches in whole genomes. Bioinformatics 19(18):2486–2488.
doi: 10.1093/bioinformatics/btg350

Contribution: design and implementation of string matching algorithm (100%), writing (90%)
WoS citations: **23** *Google Scholar citations:* **49**

As explained in the previous chapter, in 2002 I was an international fellow at the University of Padova looking for ways to improve our ability to simulate the PCR reaction products from genomic templates. Among other components, I was looking for a way to improve string matching tools for identification of short sequences in genomes. The largest genome at that time was the human reference genome with 3 billion nucleotide bases and the tool of choice for biologists was to search such sequences using a very popular software called BLAST (Altschul et al., 1990). It encapsulated a heuristic search approach using candidate sequence filtering based on short approximate hits (similarities between the search sequence and its targets). A small set of candidate hits was then explored using traditional dynamic programming algorithms for local sequence similarity to give a final answer (Galisson, 2000). While using this software as a module in Virtual PCR, I realized the approach was inefficient for repeated detection of extremely short hits in a genome or other static sequence database. I was looking for algorithms that could bring better performance to sequence comparison.

One such algorithm was used at the time in software tools called BLAT (Kent, 2002) and SSAHA (Ning et al, 2001). The algorithms reduced a genomic sequence or database to non-overlapping k-mers, used these to create a lookup/hashing table and used it to quickly cover any searched string with the k-mers and their positions in the genome. In case of a match, the string was kept as a candidate and evaluated further. However the short sequences used as PCR primers made these tools ill-tuned for string matching in the context of Virtual PCR, mainly because of the use of k-mer sizes that were two big. Upon reading some “stringology” texts and also realizing that BLAST software filtering based on a pair of hits instead of a single hit performs much better, I modified the k-mer lookup/hashing algorithmic approach to work also with two approximate hits, and further tailored the implementation to allow for easy bit-encoding of bases for the envisioned k-mer length of 8-12 (BLAT and SSAHA

used slightly longer k-mers). PRIMEX was written in C++ and had three key parameters to be set before use, *k* – the k-mer length, *m1* - the maximal number of mismatches allowed between the query k-mer and the target in the filtration step, and *m2* – the maximal desired number of mismatches to be reported for query strings (Figure 2). Using a fixed k-mer length *k* allowed the use of fast pointer arithmetics and offsets in the code. Together with encoding nucleotides into two bits this provided for compact and fast string manipulation. The second parameter, *m1* was typically 0 or 1 when used in Virtual PCR. I derived a formula based on *k*, *m1* and *m2* that can be used to determine whether the search for sequences was partly heuristic and therefore necessarily incomplete or whether the search guaranteed to find the complete set of matching sequences (Figure 2).

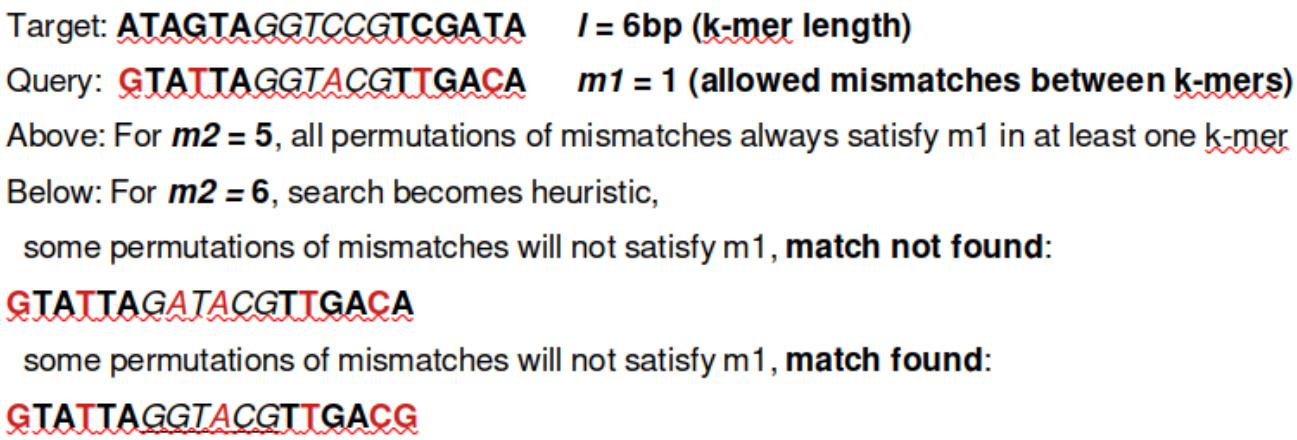

Figure 2. PRIMEX string matching software and parameter value combination thresholds resulting in a heuristic search.

To use PRIMEX with Virtual PCR efficiently another feature was needed. Calculating the lookup table from the genome was a time consuming process that took about 15 minutes for the human genome on a contemporary single desktop computer. I therefore introduced the possibility to save the lookup table to disk, allowing the user to skip this process when repeatedly searching the same sequence database or genome. Today, modern sequence alignment software always allows the creation of such index and saving of the index to disk (Langmead and Salzberg, 2012). However, it is quite possible that at the time, PRIMEX was the only software with such functionality.

To support the kind of real-time response Virtual PCR required, even saving and recovering the index from disk was not enough of a speed-up. A meaningful integration of the two programs for real-time use became only possible after I allowed PRIMEX to run in server mode, listening to commands via a port accessible locally or via the internet. A client software written in Perl took care of issuing commands on the user side. As a result, tens of minutes of BLAST search was reduced to about a minute using standalone PRIMEX and to seconds or fractions of a second using the server feature (see Table 1 in the paper). One important factor in making this approach possible and successful was the availability of GB-sized computer RAM capacity. The uncompressed index (circumventing compression was important for speed) for the human genome was on the order of 4-12GB and the speed of the server mode relied on the ability to store the entire lookup table in the RAM of the server computer and consult it as needed for rapid filtering of k-mer matches.

Using the hardware at the University of Padova I installed servers for several commonly used genomes that could be accessed from any place in the world with internet access and the PRIMEX client software running (a simple Perl script using an ad-hoc data exchange protocol). The use of this functionality did not gain much traction, probably because of the inflow of new programs from genomic teams in need of short string matching in the context of next-generation sequencing (NGS). It was mostly used via my Padova website providing sequence search support for Virtual PCR simulations. These were also hosted on a webserver in Padova and at one point enjoyed dozens of users each day. The use of the server slowly decreased towards the end of the decade and I stopped updating and supporting the software around the year 2010. The source code for both, Virtual PCR and PRIMEX are available for download from my account on Research Gate.

Another interesting fact about the PRIMEX software is, that together with BLAT and SSAHA, they were essentially the first short-read aligners, a kind of software that only enjoyed a boom few years later with the oncoming revolution in DNA sequencing (so-called NGS, or next-generation sequencing). While early NGS mapping software used similar algorithmic principles (Li et al., 2008), the use of suffix trees, suffix arrays and a specialized Burrows-Wheeler transformation-based lookup data structures, such as the FM-index (Ferragina and Manzini, 2005) came into use and superseded the previously published and used software (Li and Durbin, 2009, Li et al., 2009). A Wikipedia page listing all short-read mapping software illustrates the timeline and capabilities quite well

(https://en.wikipedia.org/wiki/List_of_sequence_alignment_software#Short-read_sequence_alignment).

Another area where string searches showed promise, was the ability to analyze proteins in a manner reminiscent of topic analysis and sentence meaning inference in computational linguistics (Lexa et al., 2009). Since biological sequences are long continuous sequences lacking any clear “punctuation symbols”, their analysis often relies on our ability to split them into meaningful subsequences. I discovered that co-ocurrence of short sequence matches determined with PRIMEX can be used to delineate protein domains (regions of proteins carrying out a conserved function, often geometrically and structurally separated from other domains of the same protein) (Lexa and Valle, 2004). My domain identification module was part of a wider software suite of a group participating in the 2004 round of CASP and CAFASP protein structure prediction competition (Jin and Dunbrack, 2005).

The few years of popularity of PRIMEX and Virtual PCR helped me to start a fruitful collaboration upon my arrival to the Faculty of Informatics at the Masaryk University in Brno. I identified a group of scientists at the Brno Technical University that was working on hardware acceleration of various computational problems. Together with Tomas Martinek (and later Ivana Burgetova, and Jiri Hon), we started an informal bioinformatics group in Brno. We published a number of papers together, two of which (covering triplex and pqsfinder software tools) are included here and mentioned in the following chapters. On the string matching front we designed several FPGA-based architectures that could speed up PRIMEX or simpler string matching tasks up to several thousand-fold (Martinek et al., 2006, Martinek et al., 2007, Martinek and Lexa, 2008, Martinek et al., 2009, Martinek and Lexa, 2010, Martinek and Lexa, 2011). The highlight of this period was a Best Poster Award from a hardware conference (Martinek and Lexa, 2011). The design of the accelerated PRIMEX and Virtual PCR was described in a conference paper presented at the 21st European Conference on Modelling and Simulation in Prague in 2007 (Lexa et al., 2007). Hardware acceleration of string matching was a direction that emerged also at other places in the world and our resources and interests would not allow us to pursue this direction further meaningfully. In the meantime several papers and commercial products and services based on FPGAs used for bioinformatics already appeared (Arram et al., 2017) and I moved on to other problems in bioinformatics in my research.

Lexa et al. 2011 - A dynamic programming algorithm for identification of triplex-forming sequences. Bioinformatics 27(18):2510-2517.

doi: 10.1093/bioinformatics/btr439

Contribution: idea for the algorithm (80%), data and implementation (20%), writing (80%)
WoS citations: **21** *Google Scholar citations*: **33**

In the year 2006, at a workshop organized by the Biophysical Institute of the Czech Academy of Sciences, I encountered a colleague, dr. Marie Brazdova studying non-B DNA structures in DNA. These are parts of the DNA molecule that have the ability to adopt a different structural organization than the typical B-DNA helix known from textbooks. Such structures have been implicated in important biological processes, including DNA replication, cancer cell life-cycle regulation, or gene regulation. The group of Marie Brazdova in Brno was studying these structures experimentally and were looking for a way to predict their formation by sequence analysis. While at that time they were mostly interested in DNA triplexes (so-called H-DNA), they were also interested in G-quadruplexes and cruciform DNA structures. This (at the time unsolved) problem immediately raised my interest. With the knowledge of a wide range of sequence analysis methods, I recognized the triplex DNA folding problem as a variant of sequence alignment, for which a dynamic-programming algorithm has existed for many years already. With a group of students who studied the problem in their theses (Kamil Rajdl, Daniel Kopecek, Juraj Jurco) and colleagues from Brno Technical University who, perhaps also under my influence, were at that time just making a shift from hardware acceleration to bioinformatics, we started studying the problem based on the above algorithmic insight.

H-DNA is made of three strands of DNA, however two of these strands form the classical Watson-Crick pair (A-T or C-G) and does not require any special computational consideration. The problem was therefore reduced to a molecular palindrome search on the remaining pair of strands with special folding rules. This basic problem has already been solved by Nussinov and Jacobson (1980) and once we found a way to split the triplex DNA formation chemical rules into eight distinct categories (Figure

3), each of which could be solved by the prediction of palindrome formation, it became clear how H-DNA structures can be predicted.

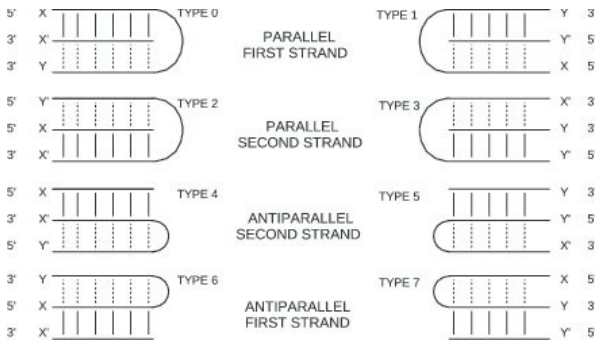

Figure 3. Eight types of triplexes that are detected in separate evaluations by the algorithm for a given region. Watson–Crick base pairing is shown by vertical bars, Hoogsteen base pairing typical for triplexes is shown with a dashed line. X and Y are two nucleotides on the same strand that will form a triplet. The eight possible triplets are: Y.X X, Y .XX , Y .X X, Y.XX , X.Y Y, X .YY , X .Y Y and X.YY (’ designates complementarity).

The major obstacle was to change the basepair formation rules from classical pairing to rules applicable to triplex formation. These rules were partly known and partly we evaluated them using molecular modelling, to calculate which basepairs are capable of forming the energetically most favorable spatial arrangements and how their geometries would support each other. While working on the problem another group from Australia published their own triplex DNA prediction algorithm called Triplexator (Buske et al., 2012). However, it was based on different principles and allowed us to soon publish also our own implementation of the above algorithm (Hon et al., 2013).

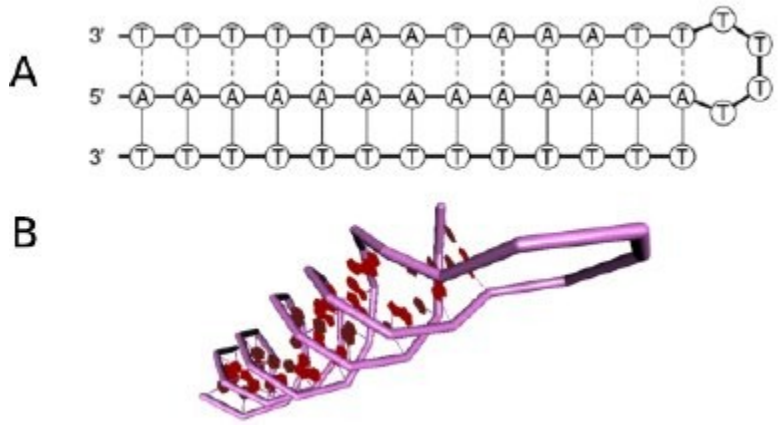

Figure 4. The triplex R/Bioconductor package. (A) 2D diagram and (B) 3D model of one of the best scored triplexes.

To motivate people to use our program, I found a bachelor student (Kamil Rajdl) at our faculty who was interested in moving our code to R/Bioconductor. At that time Bioconductor was a repository of bioinformatics software with growing popularity. He packaged our implementation into an R/Bioconductor package that allowed users to easily anayze single sequences or entire sets (Figure 4). Since the publication of Triplex, more than 7000 individual IP addresses downloaded the software from the repository (http://bioconductor.org/packages/stats/bioc/triplex/).

Later, the software was further improved with important contributions from two Faculty of Informatics students under my supervision, especially Daniel Kopecek and his thesis titled “Extension and optimization of a program for triplex detection in DNA sequences”.

Hon et al. 2017 - pqsfinder: an exhaustive and imperfection-tolerant search tool for potential quadruplex-forming sequences in R. Bioinformatics 33(21):3373–3379.
doi: 10.1093/bioinformatics/btx413

Contribution: idea for the algorithm (40%), data and implementation (10%), writing (80%)
WoS citations: **54** *Google Scholar citations*: **84**

Creating and publishing pqsfinder was a natural extension of my collaboration with the bioinformatics group at the Brno Technical University. We have been studying different non-B DNA structures for a while by then. Upon the success of triplex prediction and its application to human genomic data (Lexa et al., 2014), interest of colleagues from the Academy of Sciences in cruciform DNA, and also the increasing popularity and importance of G-quadruplexes (Yatsunyk et al., 2012), we soon started exploring the need and possibility to identify and evaluate their sequences algorithmically as well.

I spent some time evaluating cruciform DNA which resulted in a conference paper showing how sequences susceptible to their formation could be recognized. The analysis showed the importance of superhelical stress in connection with these structures. Their formation has the capacity to increase or decrease superhelicity of a circular fragment or a piece of DNA stretched between binding proteins, such as nucleosomes, scaffolding proteins, other chromatin components, transcription factors or membranes (Lexa et al., 2012).

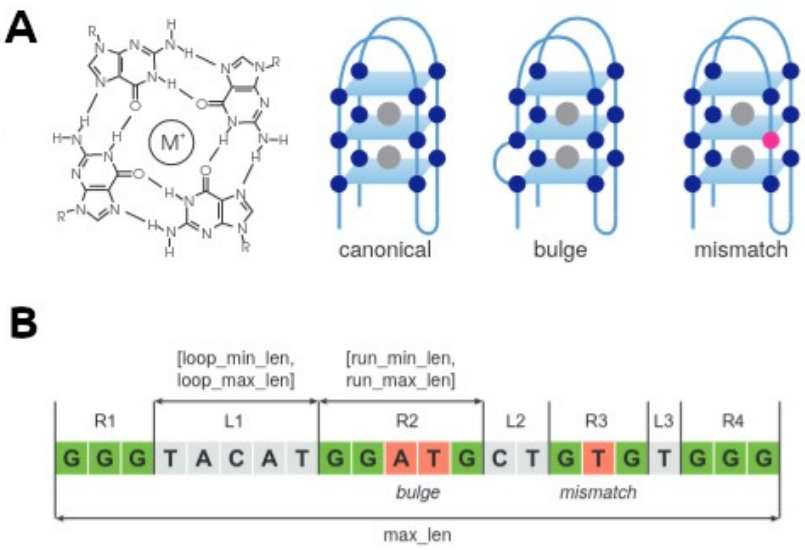

Figure 5. (A) Structural aspects of G-quadruplexes (G4s) – a planar tetrad of guanines; several tetrads can form a canonical G4, imperfect structures are possible; (B) – algorithmically important search parameters.

However, our main research direction was to be the prediction of G-quadruplex formation. At the time we encountered this problem, most researchers were combing DNA sequences using regular expressions. G-quadruplexes form in loci with several guanines clustered together and a regular expression of the form G_3-N_{1-7}-G_3-N_{1-7}-G_3-N_{1-7}-G_3 (where N is any of the four nucleotides found in DNA) would find the most typical sequences capable of G-quadruplex formation. Experimental evidence in literature pointed towards the possibility that other sequences are able to form the same structure, mostly by tolerating an imperfection or two (Mukundan and Phan, 2013), or by folding differently in space (Figure 5A). While some groups addressed this problem successfully by ignoring the precise mechanism of folding (Bedrat et al., 2016), we felt that an algorithm evaluating all possible ways to form a G-quadruplex might be feasible[3]. Jiri Hon, a collaborator of mine was most instrumental in finding the best recursion to use in the implementation that can visit all interesting combinations of guanines (G), however, for the sake of efficiency will stop that particular evaluation at a point where the prospective G-quadruplex can not result in a better evaluation score than another that

3 Interestingly, B.Subramanian (a well-known researcher in G4 studies) once remarked at a conference that such evaluation would be computationally too complex, when at the time we already had first working implementation

has already been identified in the same region (Figure 5B). Once the algorithm was fully designed and implemented, I realized we could use the results of a freshly introduced sequencing technique, called G4-seq (Chambers et al., 2015) to fine-tune and validate our program. With the help of another colleague, Tomas Martinek, using genetic programming as an optimization tool on this data, we trained the model and identified a combination of scoring parameters that gave excellent results. Not only were we able to predict G4-formation from sequence data with high accuracy, we could do so better than any of the 5 or so software tools used at that time. The accuracy of the various software tools and pqsfinder was evaluated on a dataset used also by the other research groups in this area, called Lit392 (Bedrat et al., 2015). It is a compilation of G4-forming sequences from literature that are supported by experimental evidence. The set also contains negative examples and therefore lends itself for this kind of use.

Upon our partial success with triplex and the growing popularity of R in bioinformatics circles, Jiri Hon steered the implementation of pqsfinder from the beginning to become an R/Bioconductor package (Figure 6), although the core of the package is written in C++. It became very popular soon. It is by far my best-cited paper, the R package has been downloaded so far by more than 10000 independent IP addresses (https://bioconductor.org/packages/stats/bioc/pqsfinder/)[4]. A recent review of software for identifying potential G-quadruplexes confirmed our own accuracy results (Lombardi and Londono-Vallejo, 2020) and placed pqsfinder among the most accurate. Recently, two new software products with the same motivation appeared, both based on machine learning approaches – Quadparser (Sahakyan et al., 2017) and PENGUINN (Klimentova et al., 2020). While there are indications that PENGUINN produces less false positives and has higher accuracy, the black-box aspect of the machine learning approaches keeps our approach with individually configurable imperfections and their scoring still attractive.

4 It should be noted, however, that these numbers include repeated downloads in separate years, possibly of new versions of the software, so that the real number of users is more likely in the higher hundreds to lower thousands

```
> pqsfinder(DNAString("TTTTGGGCGGGAGGAGTGGAGTTTGGGAGGGTGGGTGGGAGAA"))
  PQS views on a 43-letter DNAString subject
subject: TTTTGGGCGGGAGGAGTGGAGTTTGGGAGGGTGGGTGGGAGAA
quadruplexes:
    start width score strand nt nb nm
[1]     5    17    33      +  3  2  0 [GGGCGGGAGGAGTGGAG]
[2]    25    15    73      +  3  0  0 [GGGAGGGTGGGTGGG]
```

Figure 6. A typical use of PQSfinder to search for DNA sequences.

Also, pqsfinder has one extra feature not seen in other similar software. Its scoring function is extensible. In the manual to the software (Hon et al, 2017) we show how this feature can be used to evaluate a different class of G-quadruplexes than the ones envisaged at inception. Only a small change in code is required to provide a new type of scoring that can, for example, evaluate interstrand G-quadruplexes, where half of the guanines come from another strand of DNA, while the sequence (the one that is analyzed) has cytosines in the regions contributing to the quadruplex (Kudlicki, 2016).

My work on non-B DNA structure prediction, especially G-quadruplexes described in this chapter, led me to another collaboration with a research group at the Biophysical Institute of the Czech Academy of Sciences in Brno. In an ad-hoc and informal research collaboration on repetitive sequences in plants, together with my now long-term collaborator, dr. Eduard Kejnovsky who studies plant repeats (many of which are mobile/transposable elements), we hypothesized that non-B DNA might play some role in their life cycle. Having just finished the work on pqsfinder, I created a collection of tens of thousands of plant transposable elements and analyzed them for potential G-quadruplex forming sequences (PQS). To our surprise, there were certain areas in these DNA sequences that had much higher probability of containing a G-quadruplex then other regions (Lexa et al., 2014). They were the regulatory regions and this finding resonated well with findings in many other organisms, where G-quadruplexes were found to be enriched in gene promoters, another regulatory region in the genome. We were able to confirm similar situation in the human genome (Lexa et al., 2014b) and lately another research group observed the same results in the pig genome, so the observation, which is not currently a well-known fact, seems to hold for most eukaryotic genomes in nature by now. Much of our later research focused on pinpointing the possible roles these G-quadruplexes could play (Kejnovsky et al., 2015). We formulated a hypothesis that transposable elements, because of their mobility in genomes,

are predisposed as ideal candidates for “genomic vehicles” in eukaryotes that would populate different areas of the genome with G-quadruplexes (Kejnovsky and Lexa, 2014). These could later appear as G-quadruplex enriched gene-regulatory sequences, such as promoters or enhancers. This function of transposable elements, in my opinion, is currently underestimated and most of the research worldwide focuses on specific examples of two classes. One appears where transposable elements were exapted (captured in a specific locus of the genome) to aid or otherwise modify the expression or function of a specific gene. Another is seen where the machinery of the host which evolved to suppress potentially dangerous repeat expansion inhibits (or silences) not only the transposable element but also some sequences nearby (in case of methylation), or genome-wide (in case of small RNA production).

Together with Eduard Kejnovsky we managed to form a small research group funded by the Czech Grant Agency and occasionally other grants as well, which now focuses on this line of research. In our transposable element studies, I recognized several areas that needed fresh bioinformatics research. These lines of research are described in detail in the following two chapters.

Lexa et al. 2020 - TE-greedy-nester: structure-based detection of LTR retrotransposons and their nesting. Bioinformatics 36(20):4991–4999.
doi: 10.1093/bioinformatics/btaa632

Contribution: design of search and classification algorithm (80%), implementation and data processing (30%), writing (80%)
WoS citations: **6** *Google Scholar citations*: **8**

Building on the established collaboration in eukaryotic transposable element research described in the previous chapter, we came across an area in genome evolution and transposable element analysis that was not well covered by suitable software tools. To a certain extent there even was no clearly described and suitable algorithm for that particular sequence analysis. When studying plant genomes, we noticed that often what looks like isolated fragments of transposable elements in the genome, are actually fragments of the same element pushed apart by a later insertion of a younger transposable element into the middle of the fragmented one. This kind of arrangement reminds us a bit of the Russian "matrioshka" dolls and is commonly designated as "nested". When studying transposable elements, one often is interested in the evolution of the genome and how individual elements were moved or copied in time or in the past. We desperately needed a tool to identify the patterns of nesting, including the order and family membership of the individual elements. While some software existed at the time, there were some downsides to its use. The best of them, TE-nest (Kronmiller and Wise, 2008) suffered from very long computation times and was therefore not suitable for inspection of entire genomes.

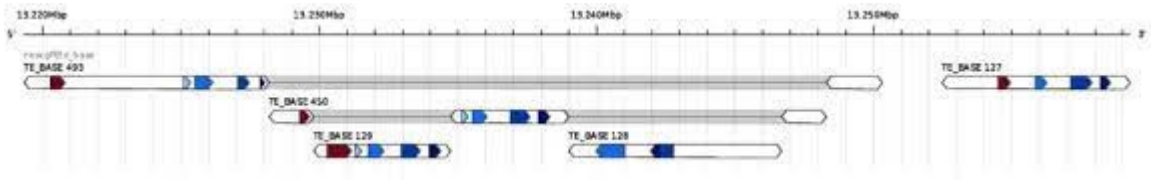

Figure 7. Nested structure of LTR retrotransposon insertion in the genome discovered and reconstructed by the TE-greedy-nester software.

After brief experiments with the idea involving a student who used the subject for their thesis (Radovan Lapar), I laid down a blueprint for a reasonable algorithmic solution. The following describes how the basic algorithm works. A greedy (not necessarily remaining such in the future, but works satisfactorily enough at the moment) iterative algorithm finds the most clearly identifiable full-length transposable elements in the analyzed DNA sequence and removes the nucleotides that are recognized as being part of the element. This, in turn may put together a previously fragmented element which now becomes recognizable as a full-length element and can be removed from the sequence by another iteration of essentially the same calculation. We quickly showed that this procedure has promise, however a series of experiments was needed to fine-tune the algorithmic steps and find the best values for a number of parameters. In the end we included a graph-based encoding for the evaluation of transposable element quality. Only elements that score well against this graph are removed as full-length elements. Another trick was to slowly lower the strictness of element identification, since the older sequences were often less conserved and removing imprecisely younger sequences from them also brought some noise into the formula. When no more transposable elements can be extracted from the analyzed sequence, the software backtracks its steps to recover the coordinates of the identified elements and outputs an annotation file in the GFF3 format that can be visualized with common genome browsing and visualization tools (Figure 7). The software was called TE-greedy-nester, with important contributions from two MU Faculty of Informatics students, Radovan Lapar, Michal Cervenansky, Jakub Horvath and Ivan Vanat. Ivan Vanat is currently getting ready to defend his master thesis bringing a number of further improvements to this procedure.

Lexa et al. 2022 - HiC-TE: a computational pipeline for Hi-C data analysis to study the role of repeat family interactions in the genome 3D organization. Bioinformatics 38(16):4030–4032.
doi: 10.1093/bioinformatics/btac442

Contribution: idea for the algorithm (100%), design and implementation of pipeline (50%), data analysis (80%), writing (80%)
WoS citations: **0** *Google Scholar citations*: **1**

In the past, molecular biology mostly studied genomes as linear arrangements of genes subject to regulatory influences among them. This was the legacy of discoveries made in studies of bacteria where linear gene structure was key to many activities of bacterial cells. However, in eukaryotes, it turns out, cellular (especially gene-regulatory) processes are more complicated (Lelli et al., 2012) and much of the regulation is not encoded as much in the linear arrangement of genomic elements but their ability to contact each other in 3D.

The latest advances in genomics allow us to ask questions about spatial arrangement of genes and other genomic DNA sequences in the nucleus of the cell (Fraser et al., 2015). In short, fragments of DNA in close proximity are fixed, joined chemically and sequenced in a paired mode in such a way that each sequencing read in the pair contains one side of the responsible contact. While the core bioinformatic support for the necessary calculation has been introduced in parallel with the experimental methods, such as 3C, 4C and HiC, the methods are mostly applicable to the non-repetitive parts of the genome. Because of my long-term interest in the repetitive fraction of the genome and bioinformatic methods that can help understand the repeats and their evolution within plant genomes, I started looking at ways to use the spatial HiC data for repeat characterization in the 3D context.

It occurred to me that where traditional HiC analysis binning procedures group HiC reads by their location (to increase the signal/noise ratio), we could try binning by repeat family classification or membership. This idea grew into a procedure that could show the most interacting repeat families in heatmaps. Together with a Faculty of Informatics student Son Hoang Nguyen, a new postdoctoral associate Monika Cechova and the transposon group of Eduard Kejnovsky, we organized the main

scripts into a robust pipeline based on the increasingly popular Nextflow workflow language. The components of the pipeline include Perl and Python string manipulation scripts, bash glue scripts and a number of R scripts for generating visualizations in the form of heatmaps and circular plots, among other types.

While most of the pipeline relies on existing software and tested visualization R packages such as circlize, karyoplotteR and ggplot, a lot of effort went into the proper normalization of counted repeat contacts. Without the normalization it would be impossible to tell which number of contacts are common/expected and which signal some kind of statistical anomaly that could represent a biologically significant phenomenon. A short description of the normalization procedures follows.

After counting all valid HiC pairs in the pipeline a table is created that contains family names in two columns (family1, family2) and in cases based on the reference genome also mapped positions (pos1, pos2). The number of combinations observed between positions and repeat families contains technical and methodological biases. For example there are many more pairs observed for adjacent positions on the same chromosomes compared to long-distance or interchromosomal HiC pairs. Some kind of normalization is therefore necessary before reporting basic statistics or creating heatmap visualizations. Choosing the right normalization method is far from trivial (Sauria et al., 2015). After careful consideration, we chose three different methods that we use in parallel towards the end of calculations in the pipeline when a family by family matrix underlying each heatmap is calculated. The three normalization methods are:

Joint probability

The baseline probability of observing a HiC pair between family A and family B is estimated as joint probability of individual probabilities for observing family A or family B in a given HiC read.

$$p(A,B) = p(A).p(B)$$

All counts of A-B pairs are divided by this number.

Label permutation

Family names in the table in columns family1 and family2 are randomly assigned to random (and therefore possibly different) rows of the table. A matrix is also created from this altered table. All counts of A-B pairs are divided by corresponding values in this matrix.

annotation shuffling

While creating the HiC pair table, a parallel table is made, which uses chromosomal positions randomly shuffled along the reference genome, while their size distribution is preserved. The pair count matrix constructed from such table is used for normalization as above (see label permutation).

As output, the pipeline generates a set of tables and images. The most informative images are the heatmaps showing normalized contacts with values differeing from 1 (Figure 8).

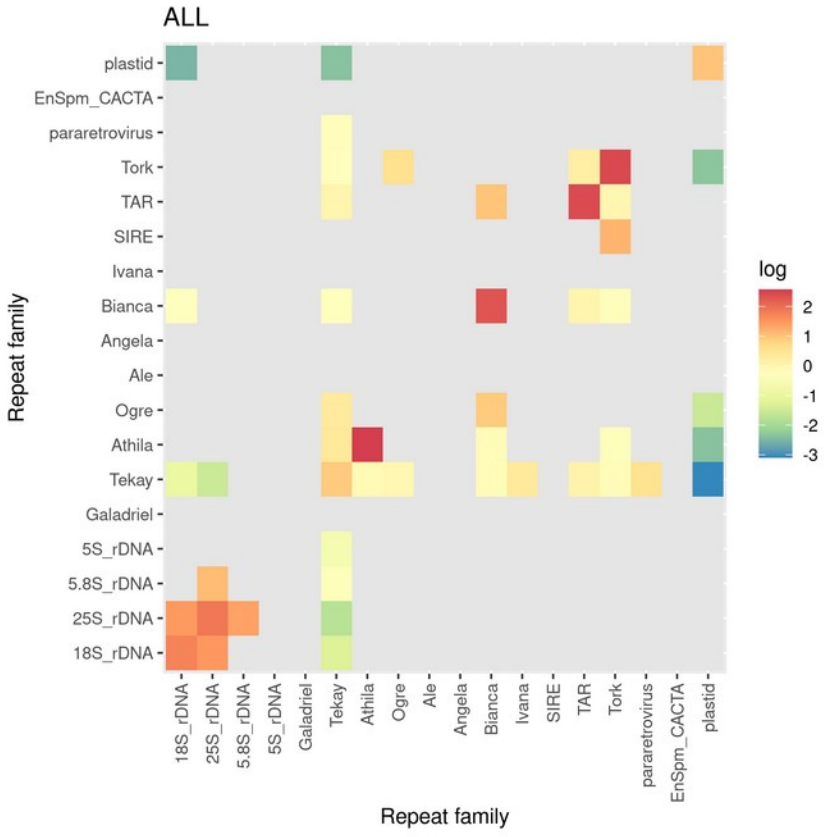

Figure 8. Heatmaps are the main output of HiC-TE allowing the user to see whether certain families deviate from the normalization expectations towards more contacts (red), or less contacts (blue) than expected.

While the pipeline will happily run on any HiC data and the corresponding reference genome, there are some limitation when running the vanilla gitlab version in such manner. Repeat classification done by Repeat Explorer (Novak et al., 2010), TE-greedy-nester and inner BLAST annotation scripts are plant-oriented, using the Neumann et al. (2019) plant TE classification scheme. TE-greedy-nester enriches the annotations for LTR retrotransposons. However, other repeat annotations may be preferred for other organisms. To make the analysis more meaningful for animal species where LTR-retrotransposons are not the main category of repeats, or to provide annotation of additional repeat classes, compared to only LTR-retrotransposons annotated by TE-greedy-nester, we allow the main reference-based repeat annotation to be provided in a GFF3 file. The pipeline is specifically tuned to accept a combination of *.out and *.gff files from RepeatMasker, but can be adapted to other external sources of annotation. The main requirement is for the GFF3 file to contain an annot="repeat_family" variable and for the corresponding Perl script (here enrich_rmsk_gff3_annotation.pl) to be able to add that name from available output.

HiC-TE pipeline is my first endeavor into the area of reproducible and scalable workflows. While back in the year 2001 it was considered satisfactory to provide a running binary or installable source code for a program to be accepted and used in biological research, nowadays the growing data volumes and increasing team work in all areas of science call for formal workflow definitions, transparent code and the use of robust frameworks that will continue working in many different use cases. HiC-TE was also our first paper published initially as a preprint in biorXiv. The experience was a positive one, with the pipeline earning its first citation while still in the preprint stage.

CONCLUSIONS

The papers discussed in this text show the importance of algorithmic approaches and data wrangling in contemporary biological research. In many of my papers, we considered a solution to a problem that others deemed impractical or even impossible to solve or improve on. Having the knowledge, access to prior research and perhaps also a bit of luck to identify a new algorithm, an old algorithm that has not been adapted to a specific class of biological data or a need among biologists that has not been met – having all this I sincerely hope I pushed our ability to analyze biological data forward. If along the way I inspired a couple of collaborators and students, I consider myself lucky, indeed.

REFERENCES

Allen, A. (2005). The Disappointment Gene. Why genetics is so far a boondoggle. https://slate.com/technology/2005/10/why-genetics-is-so-far-a-boondoggle.html Visited on Oct 22, 2022.

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J. (1990). Basic local alignment search tool. *J. Mol. Biol.* **215**:403-410.

Andreson, R., Reppo, E., Kaplinski, L. *et al.* GENOMEMASKER package for designing unique genomic PCR primers. *BMC Bioinformatics* **7**, 172 (2006). https://doi.org/10.1186/1471-2105-7-172.

Arram, J., Kaplan, T., Luk, W., Jiang, P. (2017). Leveraging FPGAs for Accelerating Short Read Alignment. *IEEE/ACM Trans Comput Biol Bioinform* **14**(3):668-677.

Atkinson, K.A. (1989). *An Introduction to Numerical Analysis (2nd ed.)*. New York: John Wiley & Sons. 693 pp.

Bedrat, A., Lacroix, L., Mergny, J.-L. (2016). Re-evaluation of G-quadruplex propensity with G4Hunter. *Nucleic Acids Research* **44**(4):1746-1759.

Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp, B.A., Wheeler, D.L. (2000). GenBank. *Nucleic Acids Research* **28**(1):15–18.

Bilofsky, H.S. and Burks, C. (1988). The GenBank ® genetic sequence data bank. *Nucleic Acids Research* **16**(5):1861–1863.

Boutros, P.C. and Okey, A.B. (2004). PUNS: transcriptomic- and genomic-in silico PCR for enhanced primer design. *Bioinformatics* **20**(15):2399-2400.

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

Chambers, V.S., Marsico, G., Boutell, J.M., Di Antonio, M., Smith, G.P., Balasubramanian, S. (2015). High-throughput sequencing of DNA G-quadruplex structures in the human genome. *Nat Biotechnol* **33**(8):877-81.

Cheeseman, J.M., Barreiro, R., Lexa, M. (1996). Plant growth modelling and the integration of shoot and root activities without communicating messengers: Opinion. *Plant and Soil* **185**(1):51-64.

Edgar, R.C., Taylor, J., Lin, V. et al. (2022). Petabase-scale sequence alignment catalyses viral discovery. *Nature* **602**, 142–147.

Edwards, M.C., Gibbs, R.A. (1994). Multiplex PCR: advantages, development, and applications. *PCR Methods Appl* **3**(4):S65-75.

Ferragina, P. and Manzini, G. (2005). Indexing Compressed Text. *Journal of the ACM*, **52**(4):553.

Fraser, J., Williamson, I., Bickmore, W.A., Dostie, J. (2015). An Overview of Genome Organization and How We Got There: from FISH to Hi-C. *Microbiol Mol Biol Rev* **79**(3):347-72.

Gagneur, J., Friedel, C., Heun, V. et al. (2017). Bioinformatics advances biology and medicine by turning big data troves into knowledge. *Informatik Spektrum* **40**, 153–160.

Galisson, F. (2000). The Fasta and Blast programs. ISMB 2000 Tutorial. url: https://www.iscb.org/cms_addon/conferences/ismb2000/tutorial_pdf/galisson5.pdf

Gröndahl, B., Puppe, W., Hoppe, A., Kühne, I., Weigl, J.A., Schmitt, H.J. (1999). Rapid identification of nine microorganisms causing acute respiratory tract infections by single-tube multiplex reverse transcription-PCR: feasibility study. *J Clin Microbiol* **37**(1):1-7.

Hon, J., Martinek, T., Rajdl, K. and Lexa M. (2013). Triplex: an R/Bioconductor package for identification and visualization of potential intramolecular triplex patterns in DNA sequences. Bioinformatics 29(15):1900-1901.

Hon, J., Lexa, M. and Martinek, T. (2017). pqsfinder: User Guide. url: https://bioconductor.org/packages/release/bioc/vignettes/pqsfinder/inst/doc/pqsfinder.html visited Oct 22, 2022.

Jin, Y. and Dunbrack, R.L. Jr. (2005). Assessment of Disorder Predictions in CASP6. *PROTEINS: Structure, Function, and Bioinformatics* **Suppl 7**:167–175.

Jung, M., Dritschilo, A. and Kasid, U (1992). Reliable and efficient direct sequencing of PCR-amplified double-stranded genomic DNA template. *Genome Res* **1**:171-174.

Kalendar, R., Khassenov, B., Ramankulov, Y., Samuilova, O., Ivanov, K.I. (2017). FastPCR: An in silico tool for fast primer and probe design and advanced sequence analysis. *Genomics* **109**(3-4):312-319.

Kejnovsky, E. and Lexa, M. (). Quadruplex-forming DNA sequences spread by retrotransposons may serve as genome regulators. *Mobile Genetic Elements* **4**(1):e28084.

Kejnovsky, E., Tokan, V. and Lexa, M. (2015). Transposable elements and G-quadruplexes. *Chromosome Research* **23**:615-623.

Kent, W.J. (2002). BLAT--the BLAST-like alignment tool. *Genome Res.* **12**(4):656-64.

Klimentova, E., Polacek, J., Simecek, P. and Alexiou, P. (2020). PENGUINN: Precise Exploration of Nuclear G-Quadruplexes Using Interpretable Neural Networks. *Front. Genet.* **27**.

Kronmiller, B.A. and Wise, R.P. (2008). TEnest: Automated Chronological Annotation and Visualization of Nested Plant Transposable Elements. *Plant Physiology* **146(**1):45–59.

Kudlicki, A.S. (2016). G-Quadruplexes Involving Both Strands of Genomic Dna Are Highly Abundant and Colocalize with Functional Sites in the Human Genome. *PLoS ONE* 11(1).

Laganière, J., Deblois, G., Lefebvre, C., Giguère, V. (2005). Location analysis of estrogen receptor α target promoters reveals that FOXA1 defines a domain of the estrogen response. *PNAS* **102**(33):11651-116.

Langmead, B., Salzberg, S. (2012). Fast gapped-read alignment with Bowtie 2. *Nature Methods* **9**:357-359.

Lelli, K.M., Slattery, M. and Mann, R.S. (2012). Disentangling the Many Layers of Eukaryotic Transcriptional Regulation. *Annual Review of Genetics* **46**:43-68.

Lexa, M. and Valle, G. (2004). Combining rapid word searches with segment-to-segment alignment for sensitive similarity detection, domain identification and structural modelling. BITS 2004 Conference, Padova, Italy. Url:http://bioinformatics.hsanmartino.it/bits_library/library/00074.pdf Visited Oct 22, 2022.

Lexa, M., Martinek, T., Beck, P., Fucik, O., Valle, G. and Zara, I. (2007). Genomic PCR simulation with hardware-accelerated approximate sequence matching. Zelinka, I., Oplatková, Z., Orsoni, A. (eds.) Proceedings 21st European Conference on Modelling and Simulation. Praha: ECMS 2007:333-338.

Lexa, M., Snášel, V., Zelinka, I. (2009). Data-Mining Protein Structure by Clustering, Segmentation and Evolutionary Algorithms. In: Abraham, A., Hassanien, AE., de Carvalho, A.P.d.L.F. (eds) *Foundations of Computational Intelligence Volume 4. Studies in Computational Intelligence* **204**:221-248.

Lexa, M., Brazdova, M., (2012). Prediction of significant cruciform structures from sequence in topologically constrained DNA - a probabilistic modelling approach. In: *Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms – BIOINFORMATICS **2012***:124-130.

Lexa, M., Cheeseman, J.M. (1995) *Simulačné modely fotosyntézy a ich vzťah k problémom vodného režimu rastlín.* In: Vedecké práce Výskumného ústavu závlahového hospodárstva v Bratislave. Bratislava: VÚZH, s. 105-117.

Lexa, M., Cheeseman, J.M. (1997). Growth and nitrogen relations in reciprocal grafts of wild-type and nitrate reductase-deficient mutants of pea (Pisum sativum L. var. Juneau). *Journal of Experimental Botany* **48**(311):1241-1250.

Lexa, M., Martinek, T. and Brazdova, M. (2014). Uneven Distribution of Potential Triplex Sequences in the Human Genome - In Silico Study using the R/Bioconductor Package Triplex. In: *Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms - BIOINFORMATICS **2014***:80-88.

Lexa, M., Kejnovsky, E., Steflova, P., Konvalinova, H., Vorlickova, M., Vyskot, B. (2014). Quadruplex-forming sequences occupy discrete regions inside plant LTR retrotransposons. *Nucleic Acids Research* **42**:968-978.

Lexa, M., Steflova, P., Martinek, T., Vorlickova, M., Vyskot, B., Kejnovsky, E. (2014b). Guanine quadruplexes are formed by specific regions of human transposable elements. *BMC Genomics* **15**(1):1032.

Li , H., and Durbin, R.M. (2009). Fast and accurate short read alignment with Burrows-Wheeler Transform. *Bioinformatics* **25**:1754-1760.

Li , R., Yu , C., Li , Y., et al. (2009). SOAP2: An improved ultrafast tool for short read alignment. *Bioinformatics* **25**:1966-1967.

Lombardi, E.P. and Londoño-Vallejo, A. (2020). A guide to computational methods for G-quadruplex prediction. *Nucleic Acids Research* **48**(1):1–15.

Martinek, T., Korenek, J., Fucik, O. and Lexa, M. (2006). A flexible technique for the automatic design of approximate string matching architectures. *IEEE Design and Diagnostics of Electronic Circuits and systems:*81-82.

Martinek, T., Fucik, O., Beck, P. and Lexa, M. (2007). Automatic generation of circuits for approximate string matching. *2007 IEEE Design and Diagnostics of Electronic Circuits and Systems:*1-6.

Martinek, T. and Lexa, M. (2008). Hardware Acceleration of Approximate Palindromes Searching. *International Conference on Field-Programmable Technology:*65-72.

Martinek, T., Vozenilek, J. and Lexa, M. (2009). Architecture model for approximate palindrome detection. *12th International Symposium on Design and Diagnostics of Electronic Circuits & Systems:*90-95.

Martinek, T. and Lexa, M. (2010). Hardware Acceleration of Approximate Tandem Repeat Detection. *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines:*79-86.

Martinek, T. and Lexa, M. (2011). Architecture model for approximate tandem repeat detection. *ASAP 2011 - 22nd IEEE International Conference on Application-specific Systems, Architectures and Processors:*239-242.

McNamee, L.M. and Ledley, F.D. (2013). Assessing the history and value of Human Genome Sciences. *Journal of Commercial Biotechnology* **19**(4):3-10.

Mukundan, V.T., Phan, A.T. (2013). Bulges in G-quadruplexes: broadening the definition of G-quadruplex-forming sequences. *J Am Chem Soc* **135**(13):5017-28.

Neumann,P., Novák, P., Hoštáková, N., Macas, J. (2019). Systematic survey of plant LTR-retrotransposons elucidates phylogenetic relationships of their polyprotein domains and provides a reference for element classification. *Mob DNA**10***:1.

Ning, Z., Cox, A.J., Mullikin, J.C. (2001). SSAHA: A Fast Search Method for Large DNA Databases. *Genome Res.* **11**:1725-1729.

Novak, P., Neumann, P., Macas, J. (2010) – Graph-based clustering and characterization of repetitive sequences in next-generation sequencing data. *BMC Bioinformatics* **11**:378.

Nussinov, R. and Jacobson, A.B. (1980). Fast algorithm for predicting the secondary structure of single-stranded RNA. *Biochemistry* **77**(11):6309-6313.

Qu, W., Shen, Z., Zhao, D., Yang, Y., Zhang, C. (2009). MFEprimer: multiple factor evaluation of the specificity of PCR primers. Bioinformatics **25**:276-278.

Rozen, S. and Skaletsky, H.J. (2000) Primer3 on the WWW for General Users and for Biologist Programmers. In: Krawetz, S. and Misener, S., Eds., *Bioinformatics Methods and Protocols: Methods in Molecular Biology*, Humana Press, Totowa, 365-386.

Rychlik, W., Rhoads, R.E. (1989). A computer program for choosing optimal oligonucleotides for filter hybridization, sequencing and in vitro amplification of DNA. *Nucleic Acids Res* **17**(21):8543-51.

Sahakyan, A.B., Chambers, V.S., Marsico, G. et al. (2017). Machine learning model for sequence-driven DNA G-quadruplex formation. *Sci Rep* **7:**14535.

SantaLucia, Jr. J. (1998). A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc Natl Acad Sci USA* **95**(4):1460-1465.

Sauria, M.E.G., Phillips-Cremins, J.E., Corces, V.G. and Taylor, J. (2015). HiFive: a tool suite for easy and efficient HiC and 5C data analysis. *Genome Biology* **16**:237.

Wang, K., Li, H., Xu, Y., Shao, Q., Yi, J., Wang, R., Cai, W., Hang, X., Zhang, C., Cai, H., Qu, W. (2019). MFEprimer-3.0: quality control for PCR primers. *Nucleic Acids Research* **47**(W1):W610–W613.

Yatsunyk, L.A., Bryan, T.M., Johnson, F.B. (2012). G-ruption: the third international meeting on G-quadruplex and G-assembly. *Biochimie* **94**(12):2475-83.