A first attempt at sequential bioinformatics

Natalie Olivia Kurtys, Johannes Gutenberg University Mainz

Introduction

In bioinformatics, a sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Very short or very similar sequences can be aligned by hand. However, most interesting problems require the alignment of lengthy, highly variable or extremely numerous sequences that cannot be aligned solely by human effort.

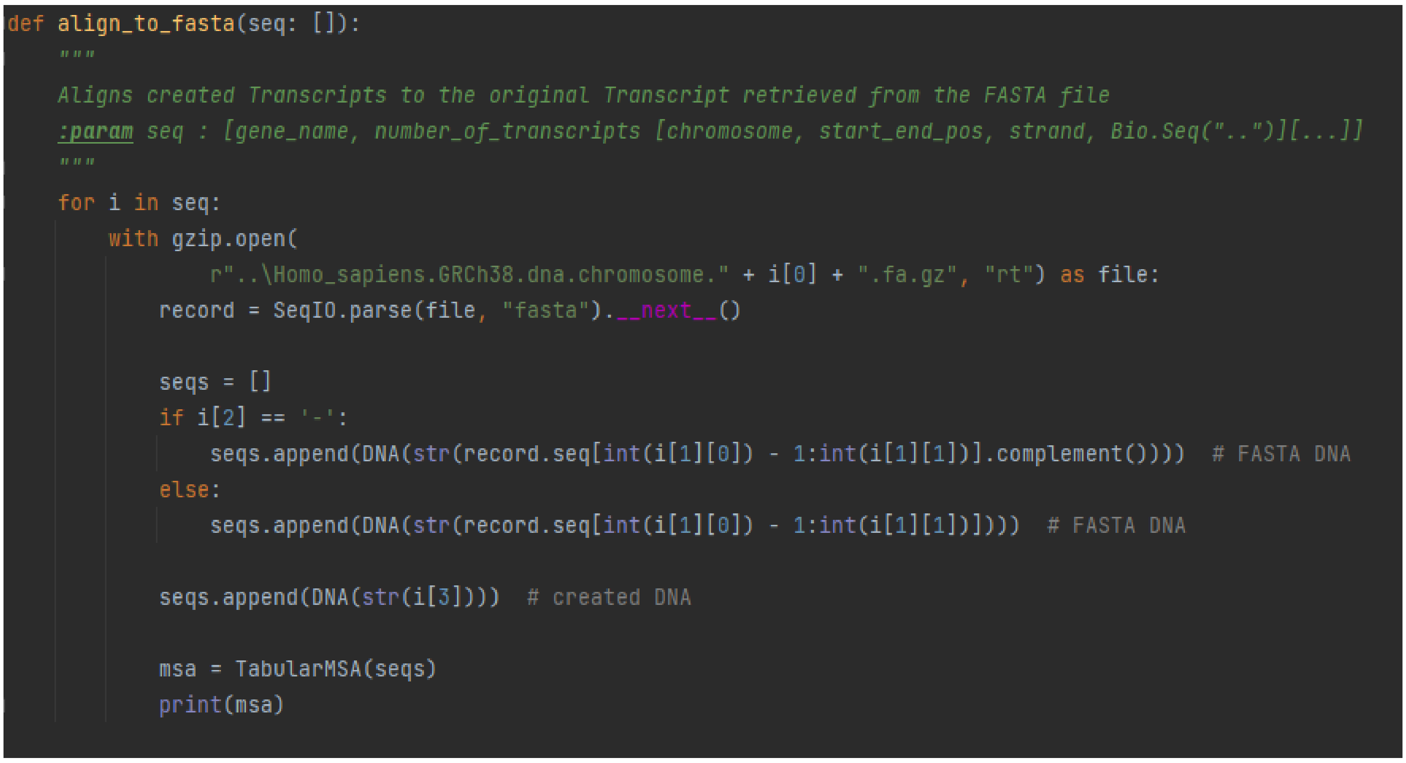
Computational approaches to sequence alignment generally fall into two categories: global alignments and local alignments. Calculating a global alignment is a form of global optimization that "forces" the alignment to span the entire length of all query sequences. By contrast, local alignments identify regions of similarity within long sequences that are often widely divergent overall. Local alignments are often preferable, but can be more difficult to calculate because of the additional challenge of identifying the regions of similarity.[1] A variety of computational algorithms have been applied to the sequence alignment problem. These include slow but formally correct methods like dynamic programming. These also include efficient, heuristic algorithms or probabilistic methods designed for large-scale database search, that do not guarantee to find best matches. There exist a variety of alignment software, all of them offering different methods to analyse and work on sequences. The goal during this project was to practice the programming skills of the student in the context of sequential bioinformatics. For that purpose, the students' task was to write code which retrieves the exons of the human genome and then adds them together so that they yield a viable transkript. This transcript can then later be used for alignments. In Addition to improving the students' skills, other soft skills such as the use of a gitlab repository and writing a ReadMe file including a documentation were practised.

To visually represent the results of the project the gene homo sapiens tetraspanin 6 (TSPAN6) will be aligned to its paralogous gene in the mouse genome. TSPAN6 is a member of the transmembrane 4 superfamily, also known as the tetraspanin family. Most of these members are cell-surface proteins that are characterized by the presence of four hydrophobic domains. The proteins mediate signal transduction events that play a role in the regulation of cell development, activation, growth and motility. The protein encoded by this gene is a cell surface glycoprotein and it functions as a negative regulator of retinoic acid-inducible gene I-like receptor-mediated immune signaling. This gene uses alternative polyadenylation sites, and multiple transcript variants result from alternative splicing.

Methods

Python was used as the primary coding language along with three python packages. PyEnsembl is a Python interface to ensembl reference genome metadata such as exons and transcripts. It is able to

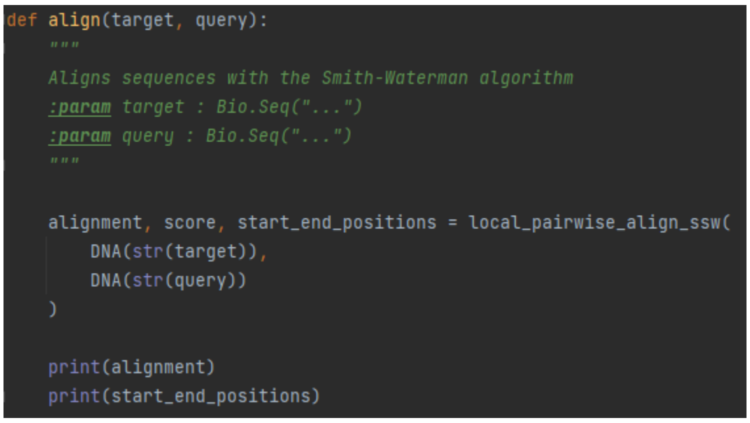
download GTF and FASTA files from the Ensembl FTP server and load them into a local database.
The project can be divided in three parts: retrieving the annotation, retrieving the sequences and adding them together as transcript and then aligning these transcripts to their paralogous transcripts. For easier overview and use the code regarding this project is also divided into three parts, which can be compared to the three parts mentioned in this document.
To retrieve the annotation for the first part of the project PyEnsembl provides a set of functions. These were used to index a GTF file containing the human genome which can be found on http://www.ensembl.org. Ensembl is a genome browser for vertebrate genomes that supports research in comparative genomics, evolution, sequence variation and transcriptional regulation. Ensembl annotates genes, computes multiple alignments, predicts regulatory function and collects disease data[2]. For this project the release 98 version of the homo sapiens genome was used.
PyEnsembl' Genome object containing the annotation allows the user to access the needed information like gene names, start and end bases of the transcripts and more. For the project the most important information was the start and end bases of exons and their respective transcripts as well as the names of the respective genes and locus'. Since the GTF file contains all various kinds of coding and non coding genes, all data was filtered to only contain protein-coding genes. After the filtering process the next step was to get information about the introns. Given only the start and end bases of exons as well as transcript, the start and end bases of the introns could be calculated. These were then sorted by start position with the exons and stored with all needed information in a file. This file will be used for the later parts of the project, which saves running time from not having to index the GTF file each time. The file follows the following pattern: each row represents one gene and is separated into three columns. The first contains information about the gene itself like name, strand, chromosome, start and end position as well as length of the gene. The second column stores the respective start and end positions as well as the length of each transcript. In the last column the exon and intron start and end positions are stored. To ease the access the exons and introns are encapsulated in brackets and are stored chronologically to the transcripts.
To be able to progress to the second part of the project a parser is needed, which parses the information out of the storage file again. With the help of the parser the information is then stored in a list following this pattern for each gene: [gene name, number of transcripts, [chromosome, start and end positions, strand, Seq("")] < repeat for every transcript >...]. The next step is to retrieve the sequences stored in FASTA files. This was done with the help of Biopython. The Biopython Project is an international association of developers of freely available Python tools for computational molecular biology and offers a FASTA file parser[4]. After retrieving and storing the sequence for each exon and intron in the aforementioned list the sequences can be put together into a transcript. To make sure that the code creates the exact transcripts, they were compared to the University of California, Santa Cruz (UCSC) Genome Database[5].
Various packages exist to create sequence alignments. Biopython offers a module for aligning sequences, however to expand the knowledge about tools in bioinformatics, scikit-bio was used in the last part of the project. Scikit-bio is a Python package providing data structures and algorithms for bioinformatics including alignment algorithms[6]. The algorithm used for this alignment was the Smith-Waterman algorithm, which performs local sequence alignment, by comparing segments of all possible lengths and optimizing the similarity measure. At first a testing alignment was performed. By cutting the real transcripts out of the human genome sequence with the help of BioPythons FASTA file parser and then aligning it to the manually created transcripts (Fig. 1). Scikit-bio's method to locally align can then be



```python
def align_to_fasta(seq: []):
    """
    Aligns created Transcripts to the original Transcript retrieved from the FASTA file
    :param seq : [gene_name, number_of_transcripts [chromosome, start_end_pos, strand, Bio.Seq("..")][...]]
    """
    for i in seq:
        with gzip.open(
                r"..\Homo_sapiens.GRCh38.dna.chromosome." + i[0] + ".fa.gz", "rt") as file:
            record = SeqIO.parse(file, "fasta").__next__()

            seqs = []
            if i[2] == '-':
                seqs.append(DNA(str(record.seq[int(i[1][0]) - 1:int(i[1][1])].complement())))  # FASTA DNA
            else:
                seqs.append(DNA(str(record.seq[int(i[1][0]) - 1:int(i[1][1])])))  # FASTA DNA

            seqs.append(DNA(str(i[3])))  # created DNA

            msa = TabularMSA(seqs)
            print(msa)
```

Fig.1: Function to align the manually created transcripts to the transcripts taken from a FASTA file. The manually created transcripts are stored in the parameter seq. For each transcript the code opens the respective FASTA file with the help of BioPythons FASTA-parser and cuts, in regard to the start and end position of the transcript the sequence and stores it as a DNA object provided by scikit-bio. Additionally the code checks on which strand the transcript is and converts if needed. The manually created transcript is always available in the form of its strand. The TabularMSA function provided by scikit-bio is the function which executes the alignment algorithm.

used to align paralogous genes as long as the sequence is being provided by a FASTA file (Fig.2).

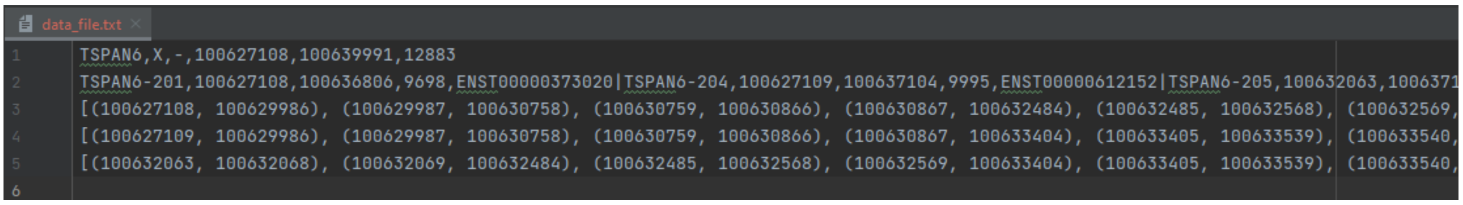


```python
def align(target, query):
    """
    Aligns sequences with the Smith-Waterman algorithm
    :param target : Bio.Seq("...")
    :param query : Bio.Seq("...")
    """

    alignment, score, start_end_positions = local_pairwise_align_ssw(
        DNA(str(target)),
        DNA(str(query))
    )

    print(alignment)
    print(start_end_positions)
```

Fig. 2: Function to align a target and a query function. Used to align paralogous genes. Both sequences are expected to be BioPythons Seq objects, which then will be translated to scikit-bios DNA objects to then execute a Smith-Waterman alignment.

Results

The results will be illustrated with the help of the gene TSPAN6. The results of the first part of the project is the file containing the protein coding genes of the human genome, in this case only the gene TSPAN6. Nonetheless the row representing this gene is very long, so for better visualization the file was adjusted as you can see in Fig. 3. Instead of tab separated columns the first line represents the gene, the second the transcripts and all following lines the exons and introns to each transcript.
The second part of the project dealt with the splicing of exons and introns from the genome and adding them up to a functioning transcript manually. In Fig. 4 it shows the alignment of the, as above mentioned, created transcript

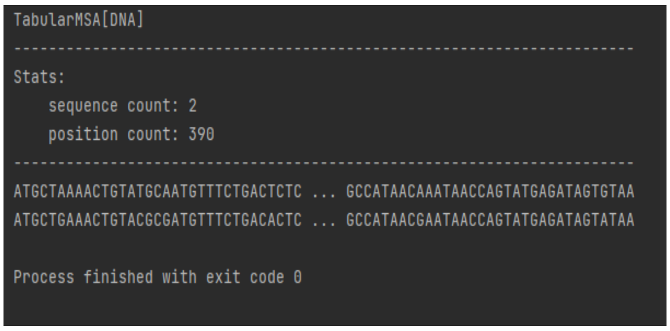


```
data_file.txt
TSPAN6,X,-,100627108,100639991,12883
TSPAN6-201,100627108,100636806,9698,ENST00000373020|TSPAN6-204,100627109,100637104,9995,ENST00000612152|TSPAN6-205,100632063,1006371
[(100627108, 100629986), (100629987, 100630758), (100630759, 100630866), (100630867, 100632484), (100632485, 100632568), (100632569,
[(100627109, 100629986), (100629987, 100630758), (100630759, 100630866), (100630867, 100633404), (100633405, 100633539), (100633540,
[(100632063, 100632068), (100632069, 100632484), (100632485, 100632568), (100632569, 100633404), (100633405, 100633539), (100633540,
```

Fig:3: Excerpt of the data file containing the annotations of the protein coding gene TSPAN 6. For easier fit into the document each “tab” was replaced by a line break

TSPAN6-204 aligned to the sequence provided by the UCSC Genome Database.

Finally the last part of the project is aligning the sequences. TSPAN 6 has got five transcripts, three of which are protein coding, one of which also appears in other species. To simplifice the results, only the alignment of this transcript is described (Identification number ENST00000612152). With the help of the written code it is possible to create an alignment with the paralogous transcripts. Given their sequences the code creates a multiFASTA file containing all sequences. This file can then be used within an alignment program to visualize the alignment outside of the terminal. In Figure 5 you can see that the alignment consists of five species: Homo sapiens, mouse, rat, dog and chimpanzee. It is visible that the chimpanzee sequence does not differ at all from the human sequence, which can be explained by the fact that humans and primates are closely related to each other. However, compared to the rat or mouse we



```
TabularMSA[DNA]
---------------------------------------------------------------------
Stats:
    sequence count: 2
    position count: 390
---------------------------------------------------------------------
ATGCTAAAACTGTATGCAATGTTTCTGACTCTC ... GCCATAACAAATAACCAGTATGAGATAGTGTAA
ATGCTGAAACTGTACGCGATGTTTCTGACACTC ... GCCATAACGAATAACCAGTATGAGATAGTATAA

Process finished with exit code 0
```

Fig.4: Alignment of the manually created transcript of TSPAN6 and the respective sequence of TSPAN6 provided by the UCSC Genome Database

have some point mutations but most of the sequence is still identical. Last but not least, the dog sequence shows most mismatches.

Discussion

The goal of this project was to practise the students' coding skills as well as expand their knowledge of available resources and tools in bioinformatics. The student practised their coding skills in the coding language python, which is widely used in bioinformatics as well as in other parts of data and computer science. Additionally they learned to organize their code according to functionality as well as to provide an easy overview over their code. With every code it is common to offer a documentation. For this project the documentation was provided in the form of a ReadMe file in Markdown. Furthermore the student learned to utilize gitlab, a repository platform similar to github which is widely used in the field of educational computer science. The student had the opportunity to learn multiple different bioinformatics python libraries during this project. This also involved reading their respective documentations and understanding them well enough to accomplish the best result in their own code. In addition, the student had to learn how to use core bioinformatic websites such as ensembl.org and the USCS Genome Database as well as commonly used file types such as FASTA and GTF.

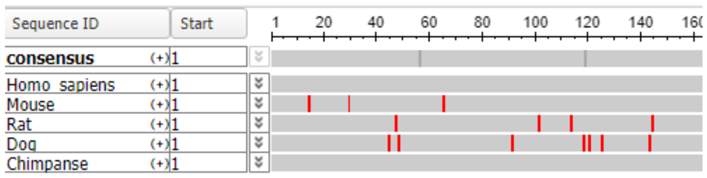


Fig.5: Multiple Sequence Alignment of five species: homo sapiens, mouse, rat, dog and chimpanzee of the transcript ENST00000612152 of the gene TSPAN6. For simplicity only bases with a 1:5 mismatch are shown. (marked red). The x axis shows the base position whereas the y axis lists the different species'.

During the process of coding it was important to plan ahead so that as few adjustments as possible have to be made to code that has already been written. In connection to that it was important to find an efficient data structure to store the information in, so it can easily be accessible. The chosen data structure was a list containing strings, lists and numbers. In retrospect it is clear that the better choice would have been to create a class containing the needed information for each gene. The decision to use a list instead caused difficulties later on in the project. Especially finding a specific gene by name or any other identification point forced an adjustment in the list, which caused all functions referring to it to be adjusted as well.

In a first attempt at sequential bioinformatics the student was able to practise with multiple commonly used tools as well expand their knowledge of sequence alignment and coding strategies. The results show functioning code which achieves the set goal. Nonetheless it can still be improved in regards to efficiency or additional functionality.

Sources

1. Polyanovsky, V. O.; Roytberg, M. A.; Tumanyan, V. G. (2011). "Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences". Algorithms for Molecular Biology. 6 (1): 25. doi:10.1186/1748-7188-6-25. PMC 3223492. PMID 22032267. S2CID 2658261.
2. Ensembl 2021. Nucleic Acids Res. 2021, vol. 49(1):884–891. PubMed PMID: 33137190. doi:10.1093/nar/gkaa942
3. PyEnsembl 2021. https://github.com/openvax/pyensembl
4. Cock, P.J.A. et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics 2009 Jun 1; 25(11) 1422-3 https://doi.org/10.1093/bioinformatics/btp163 pmid:19304878
5. UCSC Genome Browser: Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, Haussler D. The human genome browser at UCSC. Genome Res. 2002 Jun;12(6):996-1006.
6. scikit-bio 2021. https://github.com/biocore/scikit-bio