Microsatellite DNA Identification Algorithm

BME160: Final Project

Immaad Mir

University of California Santa Cruz, CA, USA
Submitted 13 March 2020

Abstract

Microsatellites are short tandem repeats of DNA which occur extremely often. Every genome has microsatellite DNA but the amount of these tandem repeats often differs from person to person. Oftentimes microsatellites are in non-coding regions of DNA but they do also occur in gene coding regions. To be able to study the effect different quantities of microsatellite repeats have on gene expression it is key we identify their location, length and amount of tandem repeats. Here we present an algorithm that locates short tandem repeats within a given file and produces a readable output file.

Keywords: Microsatellite, DNA, Genotyping, Bioinformatics, Motif, Repeat

Introduction

The majority of Eukaryotic genomes are composed of repetitive DNA segments. In Humans around 66%-69% of the genome consists of these repeats. Satellite DNA is among one of the most prominent classes of repeats and consists of motifs occurring repeatedly in tandem [4]. Depending on the length of the repeat motif they can be once again categorized; one of the most prominent types being microsatellites.

Microsatellites, also known as simple sequence Repeats (SSR), simple sequence length polymorphisms (SSLP), or short tandem repeats (STR), are satellites that range in length from one to ten (or even more) base pairs and repeat typically 5 to 50 times (This definition varies between different literatures) [2].

Due to replication slippage, where DNA strands misalign during meiosis, the quantity of repeat motifs in a given location differs from person to person. This can be helpful for genotyping and DNA fingerprinting but this can also lead to loss of protein function [3]. Diseases such as Huntingtons are caused by these discrepancies in microsatellite regions [1]. To be able to study the effect

microsatellites have on gene function it is important we locate them and identify how many of the repeats there are in tandem in certain locations.

Here we present an algorithm to locate all occurrences of microsatellite DNA within a given DNA sequence.

Methods

Our method is written in Python3, and was developed with real-world test data: A FASTA file of the latest and most polished centromere X. The test data was sequenced using Oxford NanoPore Sequencing technology and is unassembled.

There are several dependencies: sys, argparse, itertools, Bio, gzip, and pandas.

It features a pipeline with two distinct stages: creating a database of all possible repeat motifs, finding repeats in the input file and printing to the output file.

There are several command line arguments:

- **-infile** (required, type = fasta file): Input file
- **-outfile** (not required, type = fasta file, default = stdout): Output file
- **-minMonomerLen** (not required, type = int, default = 1): Minimum length of repeat motif.
- **-maxMonomerLen** (not required, type = int, default = 6): Maximum length of repeat motif.
- **-minTotLen** (not required, type = int, default = 40): Minimum length of all repeat motifs to be classed as microsatellite.

Repeat Database

To begin, all combinations of DNA nucleotides (A, T, G, C) between lengths "-minMonomerLen" and "-maxMonomerLen" are created. Each of these combinations is a repeat motif that could be found to be a microsatellite repeat monomer. Each combination is extended to less or equal to -minTotLen and stored. This is used to find microsatellites in a single iteration over the input file via comparison. If two monomers have the same repeated monomer, one is discarded.

As DNA is double stranded if there are microsatellites on one strand there must also be microsatellites on the complementary strand. The monomers are put through a function that determines which strand the microsatellites are on. This information is stored with the monomers.

Finding Microsatellite Repeats and Outputting

Given an input file if the file is zipped, gzip() is used to unzip the file. seqIQ.parse() is then used to separate the FASTA headers from the sequence. Algorithm method writeToFile() is run on each of the sequences. writeToFile() extracts substrings from the sequence and compares them to the simulated microsatellites that were found in step one. If the substring is an element of the microsatellite set- it is extended base by base until it is no longer a continuation of the identified repeat motif. The motif of the microsatellite is then printed alongside the fasta sequence it came from, the start index of the repeats, the stop index of the repeats, the repeat length, the amount of times the motif is repeated and the strand the microsatellite repeats are on. Python module Pandas is used to print output in legible aligned format.

Example:

Fasta ID	Monomer	Start	stop	length	# repeats	strand
>Test	ATGA	310	326	16	4	(+/-)

Discussion & Results

As a result of creating a simulated dictionary of repeat monomers and a simulated set of microsatellite DNA we are able to search for microsatellites with only one iteration through the input sequence. This allows, what could be a very computationally intensive, to be done in an efficient fashion.

When run on nanopore human X-chromosome reads with -minMonomerLen = 1, -maxMonomerLen = 6, -minTotLen = 40, it was found that the longest microsatellites were:

Fasta ID	Monomer	Start	stop	length	# repeats	strand
c2...	TTTC	242396	242463	67	16	-
2a...	AAATT	407511	407569	58	11	+
6d...	CTTT	212698	212750	52	13	-
...	...	...	...	...	...	...

The results show the longest microsatellite was 67 bases and consisted of the 16 repetitions of the motif TTTC. If another X-chromosome was sequenced it would be expected that in the same location there would be a similar repeat motif but repeated more or less times.

For now this algorithm can be used to analyze one data set at a time. With more time this algorithm could be run on two data sets and then compared, which would allow us to locate microsatellites which could be costly to protein function.

In addition if the chromosome was assembled, which may be possible in the near future with many new up and coming next generation sequencing devices, it is possible that with some alteration the algorithm could be used to locate whole regions of DNA such as the centromeric region which consists of, the larger, alpha satellite repeats.

Conclusion

We present a method for locating microsatellite repeats, written in Python3, that is intended to be fast and memory-efficient. Such a program would be valuable for its ability to allow researchers to find, analyze, and compare microsatellites between genomes.

Acknowledgements

Thanks to David Burnick for instructing this course, and to Karen Miga for the test data.

This algorithm is based off of several algorithms proposed by literatures. The most prominent being PERF [5].

References

1. Brouwer, Judith R et al. “Microsatellite repeat instability and neurological disease.” *BioEssays : news and reviews in molecular, cellular and developmental biology* vol. 31,1 (2009): 71-83. doi:10.1002/bies.080122
2. Garrido-Ramos, Manuel A. “Satellite DNA: An Evolving Topic.” *Genes* vol. 8,9 230. 18 Sep. 2017, doi:10.3390/genes8090230
3. *Leclercq, Sébastien et al. “DNA slippage occurs at microsatellite loci without minimal threshold length in humans: a comparative genomic approach.” Genome biology and evolution vol. 2 325-35. 12 Jul. 2010, doi:10.1093/gbe/evq023*
4. Garrido-Ramos, Manuel A. “Satellite DNA: An Evolving Topic.” *Genes* vol. 8,9 230. 18 Sep. 2017, doi:10.3390/genes8090230
5. Akshay Kumar Avvaru, Divya Tej Sowpati, Rakesh Kumar Mishra, PERF: an exhaustive algorithm for ultra-fast and efficient identification of microsatellites from large DNA sequences, *Bioinformatics*, Volume 34, Issue 6, 15 March 2018, Pages 943–948,https://doi.org/10.1093/bioinformatics/btx72