Andres Candido
ISC 4420

Final project - Written report

Text alignment program

Actual Physical DNA can be hard to manage and compare. Because of this, DNA sequences are translated into text (ATGC) which can be easily viewed, stored, and manipulated through the use of computers. This greatly simplifies many tasks that scientists might want to perform on genomic data. However, translating DNA sequences into text also has some beneficial consequences when it comes to alignment algorithms. Since DNA sequences are translated into text (ATGC), making a program that can align any form of text should follow the same logic as one that aligns DNA sequences. This means that: As long as the data given for alignment is analogous to reads in DNA sequencing, we can make a program that can align DNA without using any actual genomic data.

How do we achieve this?

We would need to replicate the process of generating DNA sequence reads while using a regular piece of text as a base sequence. Once these text “reads” are generated, alignment should follow the same logic as DNA sequence alignment. Shown in:

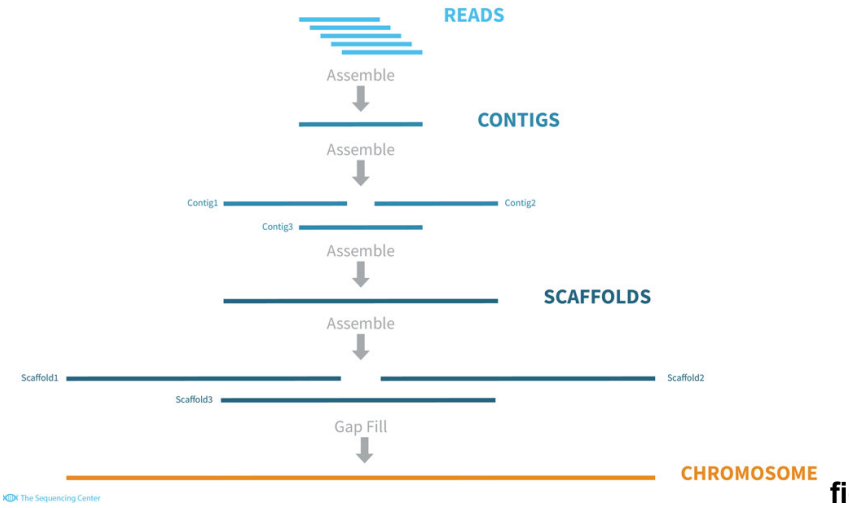
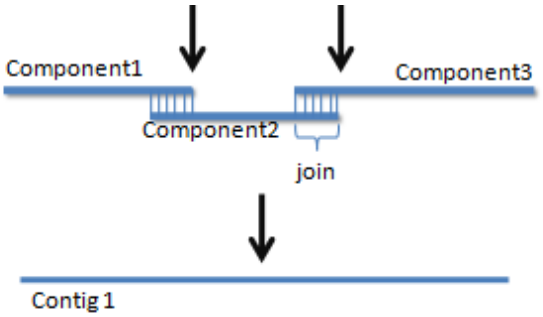


fig.1

In order to create these reads the program generates two random numbers from 1 to n, where n is the number of characters in the base sequence, and uses the smallest of the two as the beginning position of the read and the largest one as the end position. The read is extracted between these positions from the base sequence. The maximum read size can also be capped at n/x rounded to the closest integer, where x is an integer of choice (x=4 was used during code testing).
The reads are then organized into a list from longest to shortest.The longest read is selected to begin the alignment process. The program then goes through all the reads and finds matches for the ends of the longest read. The longest read and its matches are then merged

into a new longest read, the old longest and its matches are deleted from the list, and the process repeats itself until no more matches can be found. It is worth noting that although they are called “reads”, the pieces of text generated by the program are more analogous to contigs since they don’t have any errors and they are aligned in the way contigs are made into scaffolds in **fig.1**.



The final result of this sequence alignment program is an estimate of the original text sequence made using the generated reads. This estimate is stored in the form of a string that is displayed in the command window (MATLAB). It is worth noting that the final estimate can sometimes be missing some characters at the tips compared to the original, this is most likely caused due to none of the generated reads of the original text including those segments.

```
>> TextAssembler

Original =

    "A Fox one day spied a beautiful bunch of ripe grapes hanging              down and looked at the grapes in disgust."

                                                  • • •

Alignment =

    "A Fox one day spied a beautiful bunch of ripe grapes hanging              down and looked at the grape"
```

Missing part here is due none of the generated reads having that part.

Developing this program has been an interesting learning experience since most of the coding I have done so far consisted on making programs centered around solving mathematical problems and not string and array manipulation. In the short timeframe given to complete this project, I was able to create a functioning program that makes use of those concepts that were once foreing to me. I am glad to say that I'm pleased with the achieved results. Regardless of this, it is clear to see that there is plenty of room for improvement in this project, namely adding a system that actually aligns the reads into contigs in the way shown in **fig.1**, this would not only allow the program to handle reads with errors but to also be closer to completely align genomic data all the way from reads to chromosomes. If the future allows it, it would be interesting to come back to this program, try to improve upon it, and turn it into an all-purpose text sequence assembler.