Bioinformatics Lab Assignment 5: Sequence Variation and Alignment in Python

By Noah Segal-Gould

Background: Biologists frequently encounter genomic data which has undergone variations via point mutations, frameshift insertion and deletion mutations, DNA crossovers, variable copy numbers, and transposon insertions. In this lab, we utilized Python code to create these variations, detect them, and then apply the multiple sequence alignment algorithm to them.

Methods: For this assignment, we created two classes: `SequenceVariation` and `SequenceVariationDetection`. Other than the constructor, `SequenceVariation` has 5 methods for each of the desired variation types: point mutations, crossovers, copy number variation additions, copy number variation subtractions, and transposon insertions. `multiplePointMutations` works by applying a random point mutation to the sequence `n` (default 10) times. `doCrossover` works by randomly picking an index between the second character and second to last character in the string sequence, then splicing the beginning of the first sequence with the end of the second sequence as well as vice versa. `doCopyNumberVariationAddition` and `doCopyNumberVariationSubtraction` utilize and modify some k-mers code used in a previous lab to find repeated sequences within k-mers and their indices so that one more subsequence can be respectively added and removed. `doTransposon` simply replaces the first two appearances of the sticky-end sequence with the entire transposon sequence. For `SequenceVariationDetection`, we modified and built upon the mutation detection code used in the mutation exercise. If the two sequences taken within its constructor are the same when they are utilized in the `detect_variation` method, an exception is raised. Finally, several tests are run to create `seq1`-`seq7` as directed by the lab assignment instructions, and then the similarity score algorithm from the textbook as well as the multiple sequence alignment algorithm from Moodle 2 are tested on those sequences.

Results: The results successfully illustrate that the code we were assigned to write functions properly. When the program is run, the classes are instantiated and the tests are printed. Because of the length of these tests, I have decided to show the results as screenshots:

```
* Testing SequenceVariation.multiplePointMutations

seq1: GCACGTATTGATTGGCCTGTACCTA
seq2: TCACGAATTCAGAGTCCAGTACCCG

seq1 and seq2 have 9 differences

* Testing SequenceVariation.doCrossover

seq1: GCACGTATTGATTGGCCTGTACCTA and seq2: TCACGAATTCAGAGTCCAGTACCCG
become
seq3: GCACGTATTGATTGGCCTGTACCCG and seq4: TCACGAATTCAGAGTCCAGTACCTA

* Testing SequenceVariation.doCopyNumberVariationAddition

Copy-number variation addition on
GCACGTATTGATTGGCCTGTACCTA
produces
GCACGTATTGATTGATTGGCCTGTACCTA

* Testing SequenceVariation.doCopyNumberVariationSubtraction

Copy-number variation subtraction on
GCACGTATTGATTGGCCTGTACCTA
produces
GCACGTATTGGCCTGTACCTA

* Testing SequenceVariation.doTransposon

Transposon variation on
GCACGTATTGATTGGCCTGTACCTA
produces
GCACGTGGTTGCACGTATTGATTGGCCTGTACCTA

* Testing detection of mutations:

Going from GCACGTATTGATTGGCCTGTACCTA to GCACGTATTGATTGGCCTGTACCCG there was a crossover mutation
Going from GCACGTATTGATTGGCCTGTACCTA to TCACGAATTCAGAGTCCAGTACCTA there was a crossover mutation
Going from GCACGTATTGATTGGCCTGTACCTA to GCACGTATTGATTGATTGGCCTGTACCTA there was a copy variation mutation
Going from GCACGTATTGATTGGCCTGTACCTA to GCACGTATTGGCCTGTACCTA there was a copy variation mutation
Going from GCACGTATTGATTGGCCTGTACCTA to GCACGTGGTTGCACGTATTGATTGGCCTGTACCTA there was a transposon insertion mutation
```

```
* Testing similarity scores:

Similarity between GCACGTATTGATTGGCCTGTACCTA and TCACGAATTCAGAGTCCAGTACCCG: -11.0
Similarity between GCACGTATTGATTGGCCTGTACCTA and GCACGTATTGATTGGCCTGTACCCG: 17.0
Similarity between GCACGTATTGATTGGCCTGTACCTA and TCACGAATTCAGAGTCCAGTACCTA: -3.0
Similarity between GCACGTATTGATTGGCCTGTACCTA and GCACGTATTGATTGATTGGCCTGTACCTA: -7.0
Similarity between GCACGTATTGATTGGCCTGTACCTA and GCACGTATTGGCCTGTACCTA: -11.0
Similarity between GCACGTATTGATTGGCCTGTACCTA and GCACGTGGTTGCACGTATTGATTGGCCTGTACCTA: -35.0

* Testing multiple sequence alignment:

0 -GCACGT-ATTG-A--T-T-G-G---CCTGTACCTA
1 T-CACGA-ATTC-A--G-A-G-T---CCAGTACCCG
2 -GCACGT-ATTG-A--T-T-G-G---CCTGTACCCG
3 T-CACGA-ATTC-A--G-A-G-T---CCAGTACCTA
4 -GCACGT-ATTG-A--T-T-GATTGGCCTGTACCTA
5 -----------GCA-CG-T--ATTGGCCTGTACCTA
6 -GCACGTGGTTGCACGTATTGATTGGCCTGTACCTA
```

Conclusion: We successfully completed steps 1 through 8 which were assigned. We were unable to confirm that our solutions for `doCopyNumberVariationAddition` and `doCopyNumberVariationSubtraction` work on all sequences other than the initial one provided, and that our method for detecting transposon insertions will work similarly consistently.

Acknowledgements: I worked with Zoe Terhune and Aaron Krapf on this assignment. We utilized code from the textbook as well as the course Moodle 2.