Handling Indels

Arpit Agarwal

06.06.2017

Google Summer of Code 2017

Ensembl (EBI) - Genes, Genomes and Variations

Forming the Insertion_Index Array and Deletion_Index Array

Matrix for Strain A

INDEX	STRAIN A
1	A
2	C
3	G
4	T
5	G

Now w.r.t strain A there is a insertion between 1 and 2

The Strain B is A + base ‘T’ at position 2 and all the other positions are shifted.

The matrix will now look like:

INDEX	STRAIN A	STRAIN B
1	A	A
2	0	**T**
3	C	C
4	G	G
5	T	T
6	G	**T**

The strain A will be shifted by 1 (the no. of bases inserted) position for all positions below the starting index of the insertion and the empty slots generated by the shift will be filled with ‘0’s. (This new matrix with insertion can be easily converted to the bitset matrix and thus the bitset reference change algorithm is applicable)

This shift will be stored in a different array (say Insertion_Index array) like this -

Insertion_Index Array

STRAIN B
(1,1)

(No column for A because it is the reference strain right now)

The (1,1) in the strain B column means there is a insertion of 1 base after position 1.

There will be a similar array (say Deletion_Index array) which will store the indexes of deletion.

The array will look like this -

Deletion_Index Array

STRAIN B
0

No data because there are no deletion in strain B w.r.t reference strain A.

NOTE: Because the Insertion_Index Array and the Deletion_Index Array are created in a top down fashion they will be automatically sorted in ascending order of index.

Now adding the data of Strain C and visualizing the final matrix:

GIVEN DATA

INDEX	STRAIN A (Reference Strain)	STRAIN B	STRAIN C
1	A	A	A
2	0	**T**	0
3	C	C	C
4	G	G	G
5	0	0	**C**
6	0	0	**C**
7	T	T	T
8	G	**A**	G

The corresponding Insertion_Index Array and the Deletion_Index Array are:

Insertion_Index Array

STRAIN B	Strain C
(1,1)	(2,4)

Deletion_Index Array

STRAIN B	Strain C
0	0

Again the deletion array will be empty because there are no deletion with respect to reference strain A.

Changing the reference strain -

Now that we have made the shift and formed the strain matrix such that it can be converted and dealt with using the bitset matrix reference change algorithm (which will also handle the SNP's) we need to handle the Insertion_Index Array and Deletion_Index Array when changing the reference from one strain to another.

Say the reference is changed from Strain A to Strain B - The Insertion_Index_Array data of Strain A can be simply transferred to the Deletion_Index Array data of strain B and the Deletion_Index Array data of Strain B to the Insertion_Index Array of Strain A.
So, it will be like :

When Strain B is the reference -

Insertion_Index Array

STRAIN A
0

Deletion_Index Array

STRAIN A
(1,1)

All the other strains data has to be compared with the data of strain B (new reference strain) in a linear fashion - and the following case arise -

- **Insertion_Index Array**
    - There is similar insertion in New Reference strain and Other strain - **No data added to new Arrays.**
    - There are extra insertions in other strains - **Extra insertion data goes to new Insertion_Index Array.**
    - There are extra insertions in New reference strain - **Extra insertion data**

goes to new Deletion_Index_Array.

- **Deletion_Index Array**
    - There is similar deletions in New Reference strain and Other strain - **No data added to new Arrays.**
    - There are extra deletions in other strains - **Extra deletion data goes to new Deletion_Index Array.**
    - There are extra deletions in New reference strain - **Extra deletion data goes to new Insertion_Index Array.**

So, After following the above mentioned procedure we'll arrive to the following arrays:

Insertion_Index Array

STRAIN B	Strain C
0	0

Deletion_Index Array

STRAIN B	Strain C
(1,1)	(2,4)

Runtime of the Algorithm:

The runtime of this reference change depends on the no. of strains having insertions and deletions - the algorithm has a O(N) runtime complexity but will fail in case of very large indels or too many indels.

Considering our final goal of visualizing 200-300 bases (600 bases considering a right and left buffer for real time scrolling of geneoverse) - the algorithm will be able to work.