Bioinformatics: Strain Differential

Name: **Arpit Agarwal**
Mobile: +91-78791764101
Email: arpit.agarwal181@gmail.com
Alternate: f2015541@pilani.bits-pilani.ac.in
Address: 128/1227 'Y' Block,
Kidwai Nagar,
Kanpur, India - 208011

Synopsis

"Across Ensembl, large quantities of data describing genomic differences between individuals, strains or populations with respect to a designated reference genome is collected. We need efficient tools to tell us the difference between any two such accessions or to quickly identify genomes containing a given mutation"

The main objective of the project is to build a scrollable web view for the gene variations after calculating the variant after a reference change. The project will deal with SNPs and Indels for both small and large datasets.

The project is divided into three major parts:

- **Preprocessing the Data**
  - Converting to Bitset Matrix.
- **Calculating the variant differences**
  - Discussing algorithm for SNPs.
  - Discussing large data processing.
  - Generating tracks for gene-size visualisations.
  - Discussing algorithm for Indels.
- **Visualisation**
  - Genoverse as the front-end genome browser.
  - Scrollable view for the genome browser.
  - Use R for generating the backend visualization.

Preprocessing the Data:

The complete genomic data is available as .vcf files for downloading through 1001Genomes and other portals. Generally the file size is very large to be handled all at once. The file will be read using the gatk-tools-java and required data will be extracted i.e one part of a chromosome, and converted to the required bitset matrix. You can read details about the bit array here where it describes the space taken by the array as

"This set data structure uses about *n*/*w* words of space, where *w* is the number of bits in each machine word. Whether the least significant bit (of the word) or the most significant bit indicates the smallest-index number is largely irrelevant, but the former tends to be preferred (on little-endian machines)."

This means that the size of the matrix will be 1/n of the total size where n is the architecture of the machine i.e 64 bit or 128 bit .

The specification of the matrix will be as follows:

- There will be five matrices, each for the four bases and one for the absent base '-'.
- The rows of the matrix would represent the base at that location and each column will represent one specific strain.
- 1 will represent the presence of that base in that location and 0 will represent its absence.

The algorithm used for converting the variant form read from the .vcf file is shown below with the help of an example:

```
R
A T(S3)
A -(S2),G(S3)
T A(S1,S2)
G C(S2)
```

This is the variant matrix obtained from the .vcf file format.

This is a sample of 4 strain x 4 base pair. There will be five 4x4 matrices for this example. Each for 'G', 'C', 'A', 'T' & '-'.

The first row of the 'G', 'C' & '-' matrix will be '0' because of the absence of them in that row.

G MATRIX

R	S1	S2	S3
0	0	0	0
-	-	-	-
-	-	-	-
-	-	-	-

C MATRIX

R	S1	S2	S3
0	0	0	0
-	-	-	-
-	-	-	-
-	-	-	-

\- MATRIX

R	S1	S2	S3
0	0	0	0
-	-	-	-
-	-	-	-
-	-	-	-

Note: '-' in the matrix means that the information will be discussed and added later.

The ‘T’ Matrix will have a 1 at S3’s location and 0 every where else.

T MATRIX

R	S1	S2	S3
0	0	0	1
–	–	–	–
–	–	–	–
–	–	–	–

A MATRIX

R	S1	S2	S3
1	0	0	1
–	–	–	–
–	–	–	–
–	–	–	–

The matrix of the base which is in the reference strain will have a 1 at reference strain position and the union of all other matrix data for other strains. By complementing all the columns of this row (except the reference strain column) we get the complete data of A base.

Similarly completing for other rows the final data will look like

A MATRIX

R	S1	S2	S3
1	0	0	1
1	0	1	1
0	1	1	0
0	0	0	0

T MATRIX

R	S1	S2	S3
0	0	0	1
0	0	0	0
1	1	1	0
0	0	0	0

G MATRIX

R	S1	S2	S3
0	0	0	0
0	0	0	0
0	0	0	0
0	0	1	0

C MATRIX

R	S1	S2	S3
0	0	0	0
0	0	0	1
1	0	0	0
1	0	1	0

- MATRIX

R	S1	S2	S3
0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	0

These bitset matrices will be in the size order of 10 GB for the complete data and hence can be easily stored on the server side hard disks will be used whenever required.

Calculating Variant Difference

Single Nucleotide Polymorphism -

There can be only 12 possibilities of variations in case of SNP's.

Suppose we have the following data:

```
R
A T(S3)
A -(S2),G(S3)
T A(S1,S2)
G C(S2)
```

Where the reference is changed from R to S1.

```
S1
A T(S3)
A -(S2),G(S3)
A T(R,S3)
G C(S2)
```

The bitset matrices for the original data have been shown above, the bitset matrices for the changed reference data can be obtained by following these steps:

- For each row there will be two cases-
  - The base in the new reference strain remains same as the base in the old reference strain.
    - *In this case complement the reference base bitset matrix's original reference strain and new reference strain. Keep the data in all other base bitset matrices constant.*
      For the given example the first row fits this case hence first row of A base matrix will change from

A MATRIX					A MATRIX			
R	S1	S2	S3		R	S1	S2	S3
1	0	0	1		0	1	0	1
1	0	1	1	TO	1	0	1	1
0	1	1	0		0	1	1	0
0	0	0	0		0	0	0	0

- The base in the reference strain changes,
  - *In this case say the reference base changes from A in R strain to T in S1 strain (as in row 3 of the given example data). Complement the A base bitset matrix keeping the R strains value constant and complement the T base bitset matrix keeping the S1 strains value constant. Keep the data in the other 3 base matrices constant.*

  The result obtained in the changes will be

T MATRIX					T MATRIX			
R	S1	S2	S3		R	S1	S2	S3
0	0	0	1		0	1	0	1
0	0	0	0	TO	0	0	0	0
1	1	1	0		1	0	0	1
0	0	0	0		0	0	0	0

AND

A MATRIX					A MATRIX			
R	S1	S2	S3		R	S1	S2	S3
1	0	0	1		0	1	0	1
1	0	1	1	TO	0	1	1	1
0	1	1	0		1	1	0	1
0	0	0	0		0	0	0	0

Note: Changes in the second row are made according to the first condition.

Applying these two cases to the complete dataset will change the data and make S1 the new reference. The new data is as follows:

A MATRIX

R	S1	S2	S3
0	1	0	1
0	1	1	1
1	1	0	1
0	0	0	0

T MATRIX

R	S1	S2	S3
0	0	0	1
0	0	0	0
1	0	0	1
0	0	0	0

G MATRIX

R	S1	S2	S3
0	0	0	0
0	0	0	1
0	0	0	0
0	1	1	0

C MATRIX

R	S1	S2	S3
0	0	0	0
0	0	0	0
0	0	0	0
0	0	1	0

\- MATRIX

R	S1	S2	S3
0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	0

Note: This data can now be cross checked with the rules defined for creating the bitset matrix.

Runtime of the Algorithm

The major runtime of the algorithm is O(N) for a N base x M strain matrix, any operation done in the row can be performed by bit shift which is O(1). In general the algorithm is pretty efficient even for larger data.

Note: Reading and writing latency hasn't been considered in this estimation.

Generating Tracks for Gene-Size Visualization:

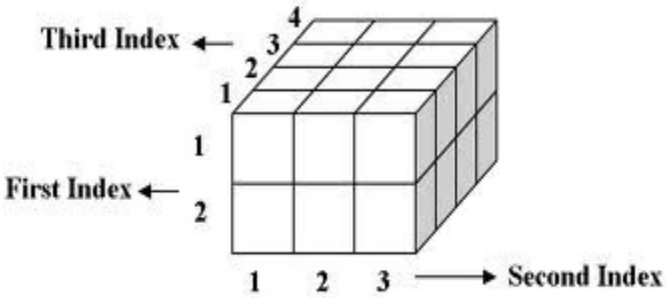
The data will be processed in small batches and tracks will be generated in standard data formats such as .vcf which can be used to visualize in any standard genome browser. This will also help in processing the scrolling buffer. The gatk-tools java which were used to convert data to bitset matrices will also be used to package the processed data into .vcf tracks.

Insertions and Deletions

The basic concept for handling insertions and deletions would be making another data structure where each element of a matrix would be an array, this array will store all the data of insertions and deletions. The array in data structure will be of variable size thus storing any insertion or deletion.

This data structure could in effect be also called a 3-D array as shown.

The first and second index correspond to the location of a specific base in the original data matrix ([1,2] is the 1st base of the second strain). The third index will store the deletions or insertions at that position, thus being variable for different positions.



Three-dimensional array with twenty four elements

This data structure can be an integer data structure or 2 bit bitset matrix (2 bits representing all the 4 bases A,T, G, C). There will be standard notation to distinguish between Insertion and Deletion.

Additionally, in case of deletion when a bunch of bases are deleted then all the 4 bitset matrices will have 0 in the position of these deleted bases thus when the reference change algorithm is run it'll show the correct difference even if that strain is the reference. For insertion the position of insertion will have a special notation in one of the 4 bitset matrices which will indicate the algorithm to process further difference for that location in the insertion matrix.

For storing the result of reference change there will be another 3D bitset matrix which will store the reference change algorithm's result for the positions at which there is an insertion or deletion.

R	S1	S2	S3
A	A	A	T
A	A	-	G
T	A	A	T
G	G	C	G

For example if A in S1 in the second row i.e. [2,2] is replaced by AA, this AA will go to the insertion data structure, the first A will be stored at [2,2,1] and the second A will be stored at [2,2,2] thus creating an array comprising of AA at the [2,2] base location. This array is of variable size and can be changed in case of further insertions or deletions.

Theoretically, this allows the storage of infinite lengths of insertion and deletions in the data structure which will be processed as required.

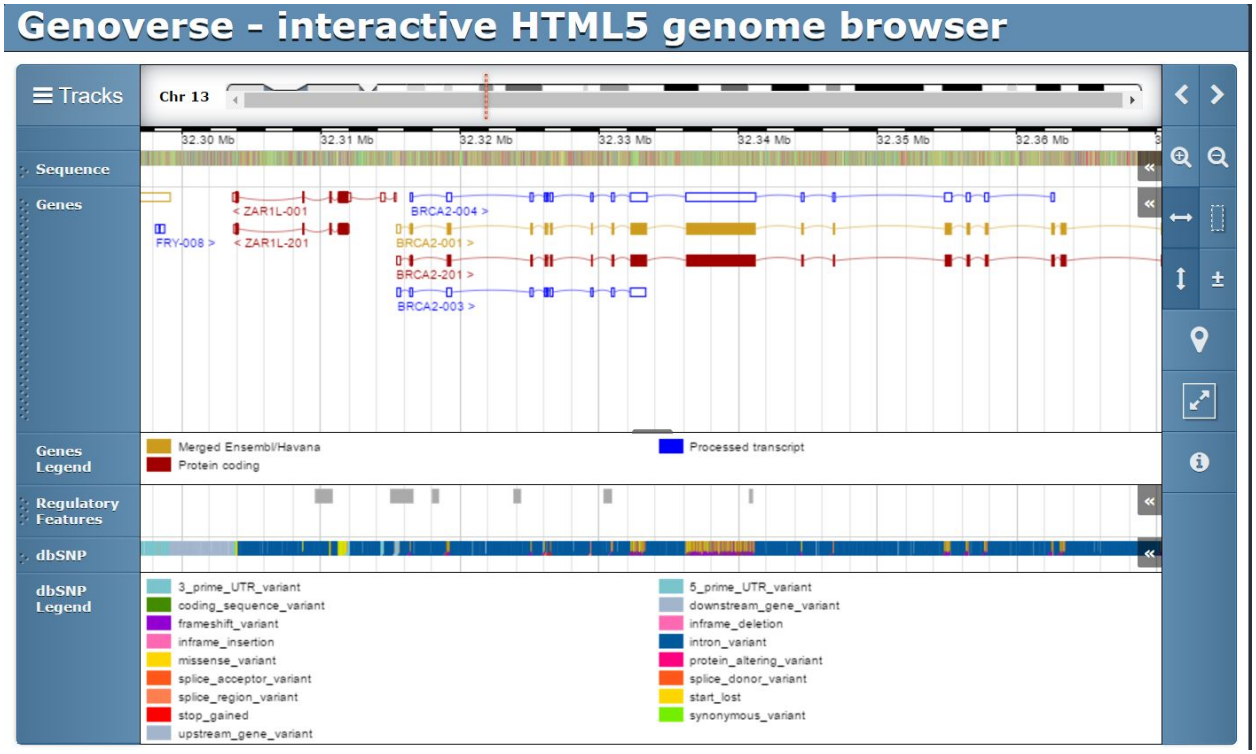
The case of insertion and deletion needs to be thought upon in much more depth to find a best optimized method. The complete period of 1 month allocated for community bonding will be spent in optimization of this case.

Visualisation

Genoverse:

The main objective of the project is to create a scrollable view of the reference change variant in a genome browser. My visualisation will be based on Genoverse, a front end JavaScript and HTML5 genome browser capable of scrollable view.

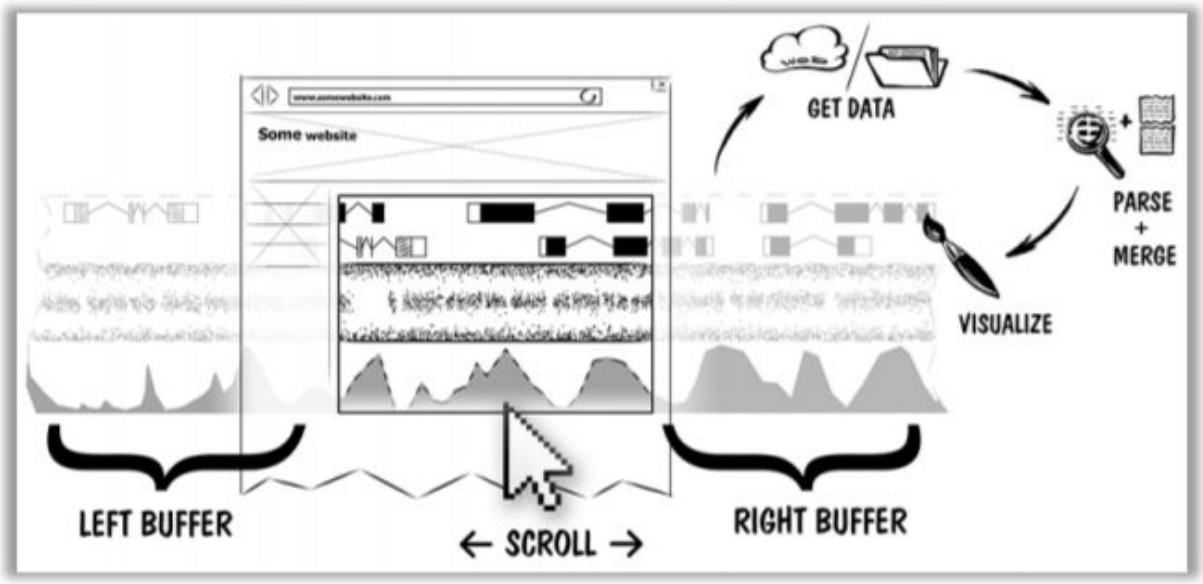
The basic view would look like this:



With various customization options available, all the options will be changed according to requirements. The main option will be to select the no. of bases to visualize.

Implementing Scrollable View:

The scrollable view will be implemented as follows:



Genoverse makes automatic AJAX calls to keep the Right and Left buffer, these AJAX calls will be replaced by backend calls to our servers and required tracks will be served from there to the web front end where it can be visualised in a scrollable web view.

The track served will be in standard file format so it can be easily read by the browser, it will be generated using the VcfR package for R.

The algorithm devised is fast and can process data in the order of 10^6 bases, this allows genoverse to process larger data file also. We require processing of at max 1000 base pairs which can be done easily in real-time for the main view and the buffer both.

Benefits to the community

Currently there are no scrollable genomic browsers which can implement reference change on the go, this project will provide scientists and researchers a tool to visualize the variants instantly helping them in analyzing the genomic variations to more depths.

This project is an intermediate project to the future goal of graphing a complete genome visualisation (many to many visualization), it deals with making the one to one visualization (one reference strain only) to one to many visualization (any reference strain).

The algorithm developed for SNPs and extended to Indels is an efficient algorithm which can be used to process huge amounts of data in very short times (10^6 base pairs within a few seconds). This algorithm can be implemented to make a RESTful API which can take large data of users and return the changed reference data.

Deliverables

The following would be the deliverables of the project:

- ➔ Processed bitset matrices.
- ➔ Implemented Backend serving variant tracks in standard file format.
- ➔ Scrollable web view using Genoverse as the frontend HTML5 JavaScript framework.

Stretch Goals

These are the objectives which are not a part of the main project but will be worked upon given extra time left at the end.

- ❑ Study Variant Effect Predictor (VEP) and its various features, get familiar with its code.
- ❑ Study and implement changes to be displayed in VEP when switching references.
- ❑ Write a RESTful API to allow processing of large scale data using the reference change algorithm.

Timeline

5th May - 30th May

- ★ Community bonding and brainstorming on improving the Indel algorithm and testing for various test cases.

30th May - 21st June

- ★ Preprocessing the data and building the required bitset matrices.

21st June - 30th June

- ★ Start implementation of the backend algorithm for generating tracks for gene-size visualizations (bitset manipulation code).

First Evaluation -

- ★ Pre processed files, work started on implementing backend algorithms.

30th June - 14th July

- ★ Finish backend implementation, start customizing Genoverse front end.
- ★ Formally define the API used by Genoverse to invoke implemented backend.

24th July - 28th July

Second Evaluation:

- ★ Implemented Backend with work started on integrating Genoverse for the front-end track visualization
- ★ Formally define the API to be used by Genoverse to call backend.

28th July - 7th August

- ★ Finish scrollable web view i.e Genoverse front end with customisations and Implemented Backend call API.
- ★ Start testing the project

7th August - 10th August

- ★ Finish testing.
- ★ Start with stretch goals.

10th August - 29th August

- ★ Try to integrate the reference change in a few features (features discussed during the community bonding period) of Variant Effect predictor tool.
- ★ Lay the groundwork for a RESTful API to process large data for reference changes.

29th August

- ★ **Final Evaluation:** submit the complete project along with some of the stretch goals.

29th August and ahead

- ★ Continue with the VEP and RESTful API goal if possible.
- ★ Contribute actively to the community.

Work done

- Downloaded and Read .vcf files for Arabidopsis using GATK command line tools.
- Extracted small sample data fromt the complete data of Arabidopsis to run tests on.
- Converted .vcf to csv and tested the reference change algorithm on small dataset.
- Evaluated various front end JavaScript visualization frameworks such as D3JS and Dygraphs for visualization of large dataset.

Requirements

As the project involves accessing and processing large amounts of data which are hosted on various servers I will require access to some kind of server to fetch as well as host my code.

Biographical Information

I am a Biology undergraduate student at the Birla Institute of Technology and Science, Pilani, India. y main focus is to combine Biology with various fields of engineering, I personally specialize in Computer Science Engineering. I want to work in the field of Bioinformatics because it is the union of my interests.

I believe that contributing to an open source project is the best way to learn development. I continue to learn from the open source community and try to add my small contributions to the community whenever possible.

I contacted Ensembl through their helpdesk email address and was connected to Benjamin Moore who further connected me to Paul Kersey, I have been in regular touch with Paul for about a month now, he has been guiding me in improving my understanding of the project and also understanding various tools and technical details that came along in the way.

I have previously worked on setting up a high performance tile server for a game studio. I am currently working on a research project where we are trying to automate the Pap Smear test for cervical cancer using Image Processing.

I am a tech enthusiast, and I particularly enjoy solving puzzles (I am an avid cuber) and reading books. I am also interested in photography which has recently got me interested in travelling.

You can view my Linkedin and GitHub profile for further information.