Understanding Sequence Comparison and Influenza Virus

Abstract

Flu is an infectious disease caused by an influenza virus and the vaccination against these viruses are less effective. In this project we are comparing Influenza A virus from different sources, A/human/Ohio/2006(H3N2) and A/Shanghai/02/2013(H7N9). Out of the 8 segments we are focusing on the segment 1 of the Influenza A virus. We have implemented Global alignment method for the sequence alignment and also used BLAST. Then we are comparing the outputs like indel, synonymous mutation and nonsynonymous mutation from both techniques. We have concluded that because of the mutations in the influenza virus it is necessary to update the influenza vaccine every year.

1.0 Introduction

Four influenza pandemics, starting with the historic 1918 pandemic, have killed thousands of people around the world[1]. The influenza A, B, and C viruses, represents three of the five genera of the family Orthomyxoviridae[2]. Out of these, Influenza A and Influenza B viruses account for most of the Influenza-related hospitalizations. Even though we were successful to study the biological structure and behavior of Influenza viruses, because of their antigenic drift and antigenic shift, makes it difficult to develop a vaccination against these viruses.

Antigenic drift are small changes in the genes of influenza viruses that happen continually over time as the virus replicates. These small genetic changes usually produce viruses that are pretty closely related to one another, which can be illustrated by their location close together on a phylogenetic tree. These viruses usually share the same antigenic properties and an immune system exposed to an similar virus will usually recognize it and respond[3]. But these small genetic changes can accumulate over time and result in viruses that are antigenically different (further away on the phylogenetic tree). When this happens, the body's immune system may not recognize those viruses.

Antigenic shift is the process by which two or more different strains of a virus, or strains of two or more different viruses, combine to form a new subtype having a mixture of the surface antigens of the two or more original strains[4]. Antigenic shift is an abrupt, major change in the influenza A viruses, resulting in new HA and/or new HA and NA proteins in influenza viruses that infect humans. Shift results in a new influenza A subtype or a virus with a HA or a HA and NA combination that has emerged from an animal population that is so different from the same subtype in humans that most people do not have immunity to the new virus. Such a "shift" occurred in the spring of 2009, when an H1N1 virus with a new combination of genes emerged to infect people and quickly spread, causing a pandemic. When shift happens, most people have little or no protection against the new virus[3].

We are focusing on the segment 1 of the influenza A virus from different sources namely, A/human/Ohio/2006(H3N2) and A/Shanghai/02/2013(H7N9). Segment alignment can be used to compare these two sequences and to find out the insertions, deletions mutations among them. We have used global alignment and BLAST alignment to compare the sequences.

Global Alignment or the Needleman–Wunsch algorithm is an algorithm used in bioinformatics to align protein or nucleotide sequences[5]. It uses dynamic programming approach to compare biological sequences. The algorithm essentially divides a large problem (e.g. the full sequence) into a series of smaller problems and uses the solutions to the smaller problems to reconstruct a solution to the larger problem. Closely related sequences which are of same length are very much appropriate for global alignment. Here, the alignment is carried out from beginning till end of the sequence to find out the best possible alignment[6]. The implementation of this algorithm is described in 3.0 Implementation of this report.

BLAST or Basic Local Alignment Search Tool finds regions of similarity between biological sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance[7]. Since we have two sequences, we are not using the database search but uses Align two or more sequences. It returns a detailed output which is described in the section 4.0 Results and Discussion.

2.0 Materials and Methods

Segment 1 of the Influenza A virus from two sources are used in this project.

a. Influenza A virus (A/human/Ohio/2006(H3N2)) segment 1 polymerase PB2[8].
   From the naming we can conclude that, this is the segment 1 of the human-isolated H3N2 influenza virus, taken in Ohio in 2006.
b. Influenza A virus (A/Shanghai/02/2013(H7N9)) segment 1 polymerase PB2[9].
   From the naming we can conclude that, this is the segment 1 of the human-isolated H7N9 influenza virus, taken from the country China in 2013.

We have used Global alignment algorithm to compare the two input sequences. This is the best alignment over the entire length of two sequences, and is suitable when the two sequences are of similar length, with a significant degree of similarity throughout.

In order to perform a Needleman-Wunsch (Global) alignment, a matrix is created which allows us to compare the two sequences. The matrix if filled depending up on the match / mismatch or gap scores. Once we have computed this score for every cell, we must do a "traceback", that is to determine the actual set of operations that lead to the score. Because when computing the score of a cell we took a max over three numbers, on the traceback we go to the location of the highest – going sideways or up corresponds to gaps, and going along the diagonal corresponds to a match.

This algorithm performs alignments with a time complexity of O (mn) and a space complexity of O (mn). Where m and n are the length of the input sequences.

BLAST is one of the most widely used bioinformatics programs for sequence searching. General Concept for Original BLAST Program are as follows

- Sequence (query) is broken into words of length W
- Align all words with sequences in the database
- Calculate score T for each word that aligns with a sequence in the database using a substitution matrix
- Discard words whose T value is below a neighborhood score threshold
- Extend words in both directions until score falls by dropoff value X when compared to previous best score

The BLAST algorithm identifies regions of local sequence similarity by first identifying candidate similar sequences that have k-tuples in common with the query sequence, and then extending the regions of similarity. Its computational complexity is O(n)[10].

3.0 Implementation

a. Global Alignment

We have implemented in Java. We used gap penalty of -2, mismatch = -1, match = 1. Data structure: two arrays of size M*N (length of two sequences). One to store highest value of the alignment and the other one (traceback) to store from which step we got the score (up, left or diagonal).

We used two main functions:

- Align: which go through both sequences and update the scores matrix as follows:

$$M(i,j) = MAX(M_{i-1,j-1} + S(A_i, B_j)$$
$$M_{i-1, j} + gap$$
$$M_{i,j-1} + gap)$$

It also updates the traceback matrix with values up, left or diagonal according to the max values we previously got.

- Traceback: that gives us the optimal alignment after traceback through the array we got and update the new sequences with corresponding values

  Up: put gap in the second sequence.
  Left: put gap in first sequence.
  Diagonal: put corresponding letters from both sequences.

b. BLAST

We have used the website[7] to compare the sequences A/human/Ohio/2006(H3N2) and A/Shanghai/02/2013(H7N9). We used Blastn and Megablast option. The result is shown in the result section.

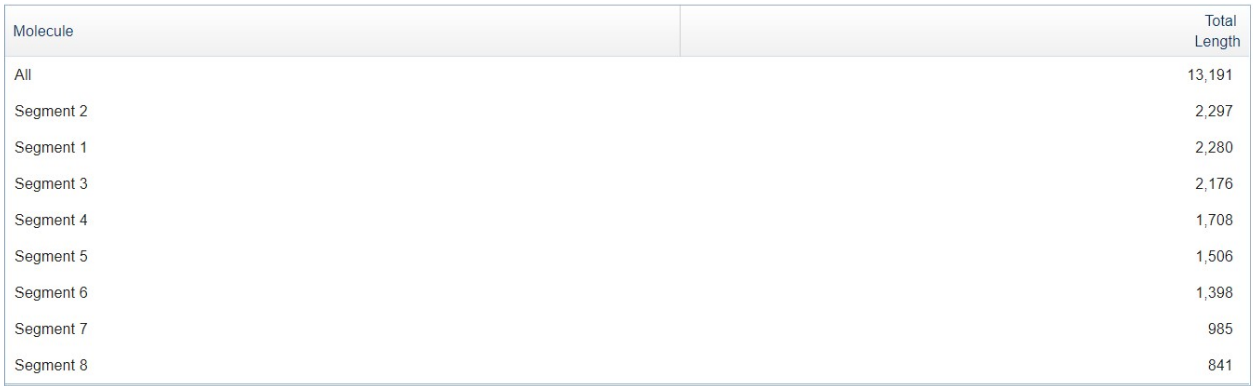
c. Mutations

We have used python and the data structure, dictionaries are used to store the codon to protein translation table. The code is in the section 7.0 Appendices.

- Read the input sequences.
- Find out the Indel count by checking whether '-'is present in the sequence or not.
- Compare both sequences to find the number of same nucleotides and different ones.
- Convert the sequence to protein sequences.
- If different codons translates to same protein, it is counted towards the synonymous mutation.
- If different codons translates to different protein, then it is a non-synonymous mutation.
- Print the outputs.

4.0 Results and Discussion

1) Understand Genbank entries at NCBI and answer the following questions for[11]:
   a. What's the size/length of this flu virus genome?
      13,191. Segment 1 has 2,280 genome.



Molecule	Total Length
All	13,191
Segment 2	2,297
Segment 1	2,280
Segment 3	2,176
Segment 4	1,708
Segment 5	1,506
Segment 6	1,398
Segment 7	985
Segment 8	841

Figure 1: Showing size of the Influenza A virus

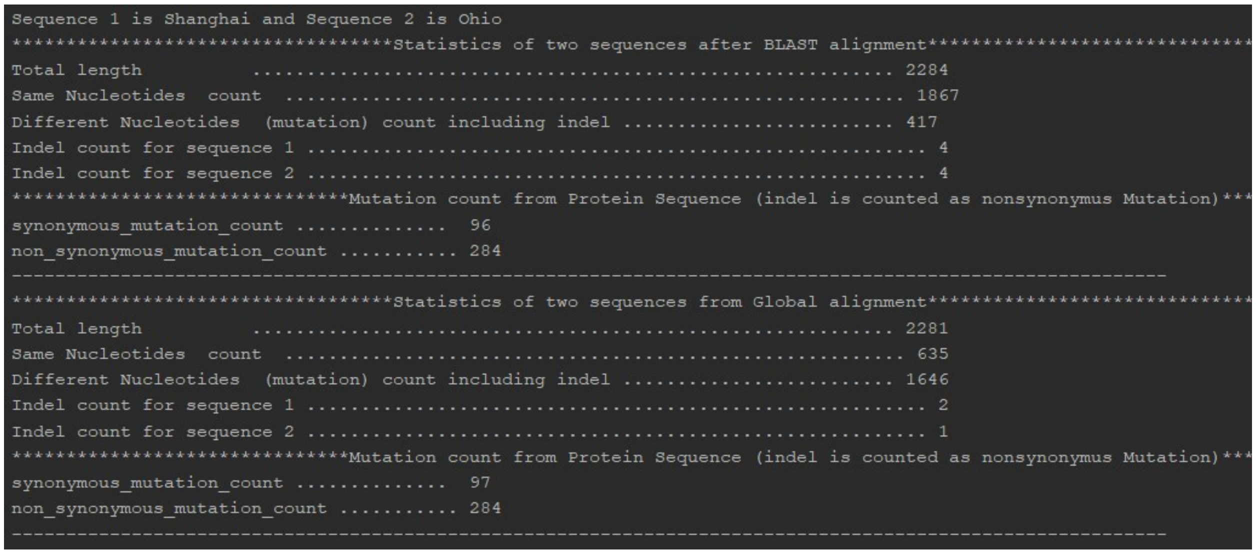
   b. What is it made of (RNA/DNA)? RNA
   c. How many genes does this virus genome contain?
      13. [12]
   d. What are their names? [13]
      1. HA - hemagglutinin
      2. M1- matrix protein 1
      3. M2 - matrix protein 2
      4. NA - neuraminidase
      5. NEP - nuclear export protein
      6. NEWENTRY - Record to support submission of GeneRIFs for a gene not in Gene.
      7. NP - nucleocapsid protein
      8. NS1 - nonstructural protein 1
      9. PA - polymerase PA
      10. PA-X - PA-X protein
      11. PB1 - polymerase PB1
      12. PB1-F2 protein
      13. PB2 - polymerase PB2 – Segment 1 gene.
   e. What does CDS mean?
      Coding Region.
   f. How many CDSs are there?
      1 for segment1
   g. How many proteins does the virus genome code?
      125[12] total for 8 segments. 1 for segment 1.
   h. What are they?
      Segment 1- polymerase PB2[14].

2) Mutations from Global Alignment

NP	#Indel	#Synonymous Mutation	#Nonsynonymous Mutation	Other things
Shanghai	2	97	284	
Ohio	1	97	284	

3) Mutations from BLAST

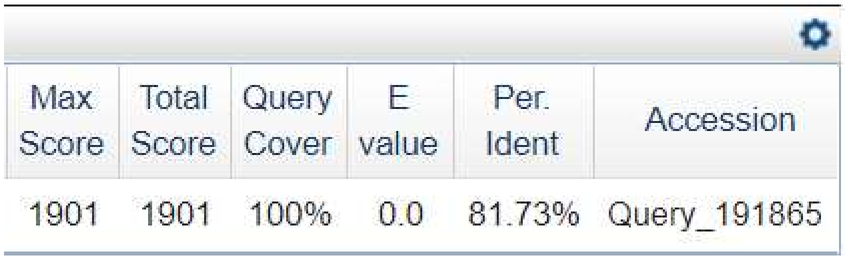
NP	#Indel	#Synonymous Mutation	#Nonsynonymous Mutation	Other things
Shanghai	2	96	284	
Ohio	2	96	284	



```
Sequence 1 is Shanghai and Sequence 2 is Ohio
**********************************Statistics of two sequences after BLAST alignment*****************************
Total length          ........................................................... 2284
Same Nucleotides  count  ......................................................... 1867
Different Nucleotides  (mutation) count including indel ......................... 417
Indel count for sequence 1 ...................................................... 4
Indel count for sequence 2 ...................................................... 4
******************************Mutation count from Protein Sequence (indel is counted as nonsynonymus Mutation)***
synonymous_mutation_count .............  96
non_synonymous_mutation_count ........... 284
------------------------------------------------------------------------------------------------------------
**********************************Statistics of two sequences from Global alignment*****************************
Total length          ........................................................... 2281
Same Nucleotides  count  ......................................................... 635
Different Nucleotides  (mutation) count including indel ......................... 1646
Indel count for sequence 1 ...................................................... 2
Indel count for sequence 2 ...................................................... 1
******************************Mutation count from Protein Sequence (indel is counted as nonsynonymus Mutation)***
synonymous_mutation_count .............  97
non_synonymous_mutation_count ........... 284
------------------------------------------------------------------------------------------------------------
```

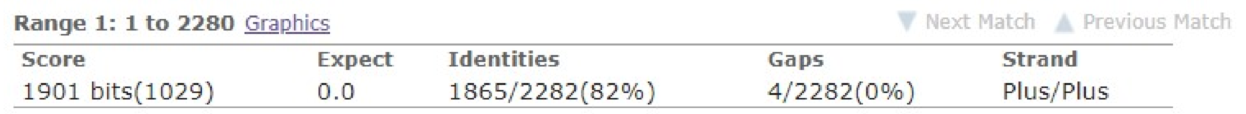
Figure 2: mutations after global alignment and BLAST

4) Result from BLAST



Max Score	Total Score	Query Cover	E value	Per. Ident	Accession
1901	1901	100%	0.0	81.73%	Query_191865

Figure 3: BLAST output

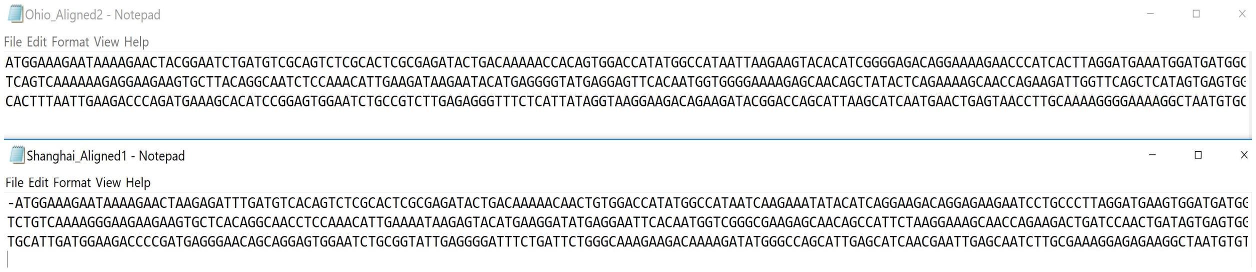


Range 1: 1 to 2280 Graphics

Score	Expect	Identities	Gaps	Strand
1901 bits(1029)	0.0	1865/2282(82%)	4/2282(0%)	Plus/Plus

Figure 4: BLAST output

5) Global Alignment Result



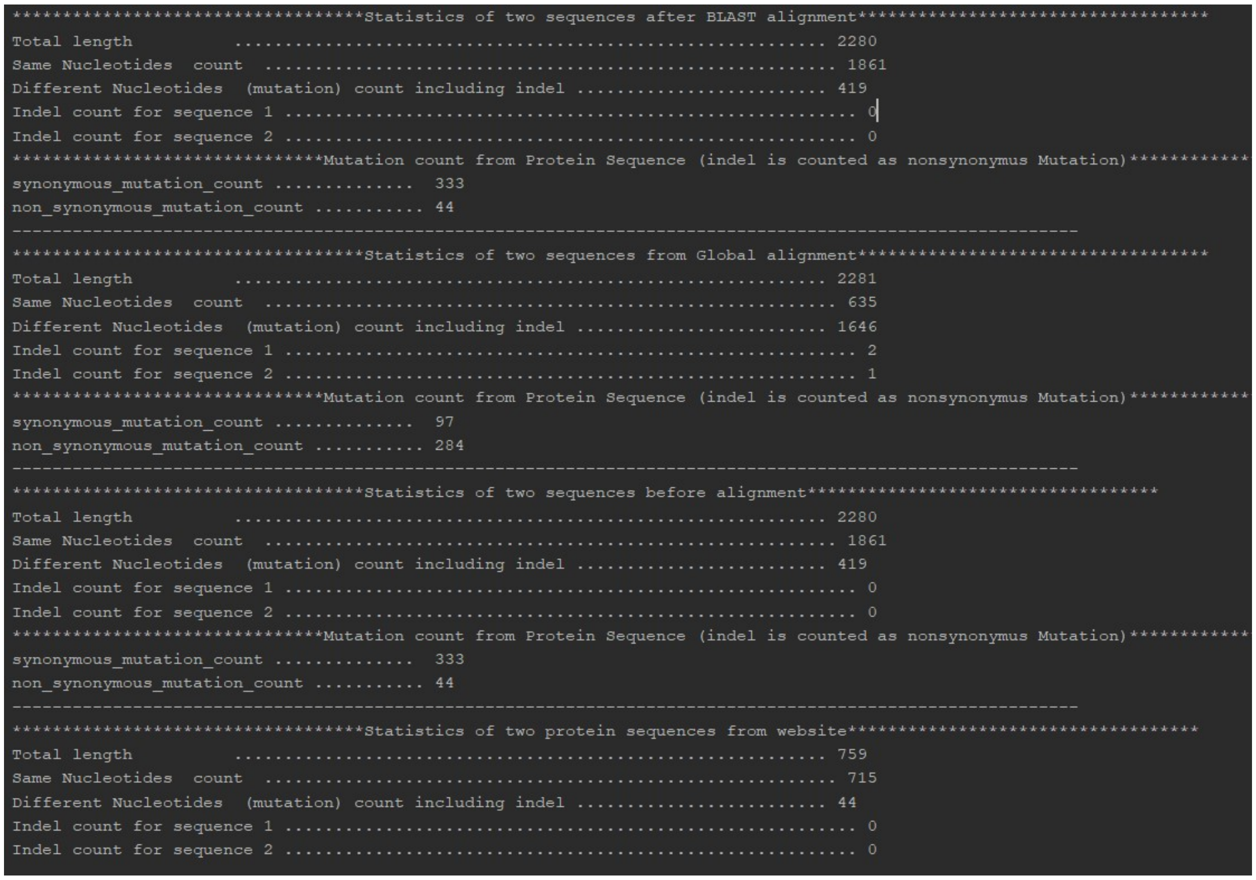
```
ATGGAAAGAATAAAAGAACTACGGAATCTGATGTCGCAGTCTCGCACTCGCGAGATACTGACAAAAACCACAGTGGACCATATGGCCATAATTAAGAAGTACACATCGGGGAGACAGGAAAAGAACCCATCACTTAGGATGAAATGGATGATGGC
TCAGTCAAAAAAGAGGAAGAAGTGCTTACAGGCAATCTCCAAACATTGAAGATAAGAATACATGAGGGGTATGAGGAGTTCACAATGGTGGGGAAAAGAGCAACAGCTATACTCAGAAAAGCAACCAGAAGATTGGTTCAGCTCATAGTGAGTGG
CACTTTAATTGAAGACCCAGATGAAAGCACATCCGGAGTGGAATCTGCCGTCTTGAGAGGGTTTCTCATTATAGGTAAGGAAGACAGAAGATACGGACCAGCATTAAGCATCAATGAACTGAGTAACCTTGCAAAAGGGGAAAAGGCTAATGTGC
```

```
-ATGGAAAGAATAAAAGAACTAAGAGATTTGATGTCACAGTCTCGCACTCGCGAGATACTGACAAAAACAACTGTGGACCATATGGCCATAATCAAGAAATATACATCAGGAAGACAGGAGAAGAATCCTGCCCTTAGGATGAAGTGGATGATGG
TCTGTCAAAAGGGAAGAAGAAGTGCTCACAGGCAACCTCCAAACATTGAAAATAAGAGTACATGAAGGATATGAGGAATTCACAATGGTCGGGCGAAGAGCAACAGCCATTCTAAGGAAAGCAACCAGAAGACTGATCCAACTGATAGTGAGTGG
TGCATTGATGGAAGACCCCGATGAGGGAACAGCAGGAGTGGAATCTGCGGTATTGAGGGGATTTCTGATTCTGGGCAAAGAAGACAAAAGATATGGGCCAGCATTGAGCATCAACGAATTGAGCAATCTTGCGAAAGGAGAGAAGGCTAATGTGT
```

Figure 5: Global Alignment output

6) Other things:

The mutation before alignment and the protein sequence from the blast site, as well as the BLAST result with the criteria: Match/Mismatch scores as 4,-5 and Gap costs as 12 and extension 8 as follows.



```
***********************************Statistics of two sequences after BLAST alignment***********************************
Total length          ........................................................... 2280
Same Nucleotides  count  ........................................................ 1861
Different Nucleotides  (mutation) count including indel ......................... 419
Indel count for sequence 1 ...................................................... 0
Indel count for sequence 2 ...................................................... 0
*******************************Mutation count from Protein Sequence (indel is counted as nonsynonymus Mutation)************
synonymous_mutation_count .............  333
non_synonymous_mutation_count ........... 44
----------------------------------------------------------------------------------------------------
***********************************Statistics of two sequences from Global alignment***********************************
Total length          ........................................................... 2281
Same Nucleotides  count  ........................................................ 635
Different Nucleotides  (mutation) count including indel ......................... 1646
Indel count for sequence 1 ...................................................... 2
Indel count for sequence 2 ...................................................... 1
*******************************Mutation count from Protein Sequence (indel is counted as nonsynonymus Mutation)************
synonymous_mutation_count .............  97
non_synonymous_mutation_count ........... 284
----------------------------------------------------------------------------------------------------
***********************************Statistics of two sequences before alignment***********************************
Total length          ........................................................... 2280
Same Nucleotides  count  ........................................................ 1861
Different Nucleotides  (mutation) count including indel ......................... 419
Indel count for sequence 1 ...................................................... 0
Indel count for sequence 2 ...................................................... 0
*******************************Mutation count from Protein Sequence (indel is counted as nonsynonymus Mutation)************
synonymous_mutation_count .............  333
non_synonymous_mutation_count ........... 44
----------------------------------------------------------------------------------------------------
***********************************Statistics of two protein sequences from website***********************************
Total length          ........................................................... 759
Same Nucleotides  count  ........................................................ 715
Different Nucleotides  (mutation) count including indel ......................... 44
Indel count for sequence 1 ...................................................... 0
Indel count for sequence 2 ...................................................... 0
```

Figure 6: Mutations with high gap penalty in BLAST

7) Observation

The presence of gap results in high non-synonymous mutation which means an indel has more effect in gene mutations.

5.0 Summary

THE BIOLOGY OF INFLUENZA VIRUSES[2]

The influenza A, B, and C viruses are characterized by segmented, negative-strand RNA genomes representing three of the five genera of the family Orthomyxoviridae. These viruses share a common genetic ancestry, but genetically diverged. The exchange of viral RNA segments between viruses – has been reported to occur within each genus, or type, but not across types. Influenza A viruses are further characterized by the subtype of their surface glycoproteins, the hemagglutinin (HA) and the neuraminidase (NA). While many genetically distinct subtypes (16) for HA and (9) for NA, it have been found in circulating influenza A viruses, only three HA (H1, H2, and H3) and two NA (N1 and N2) subtypes have caused human epidemics, as defined by sustained, widespread, person-to-person transmission.

The organization of the influenza B virion is similar to influenza A, with four envelope proteins: HA, NA, and, instead of M2, NB and BM2. Influenza C virions are structurally distinct from those of the A and B viruses. The influenza A and B virus genomes each comprise eight negative-sense, single-stranded viral RNA (vRNA) segments, while influenza C virus has a seven-segment genome. The eight segments of influenza A and B viruses (and the seven segments of influenza C virus) are numbered in order of decreasing length. The genomic organization of influenza C viruses is generally similar to that of influenza A and B viruses; however, the HEF protein of influenza C replaces the HA and NA proteins, and thus the influenza C virus genome has one fewer segment than that of influenza A or B viruses.

A segmented genome enables antigenic shift, in which an influenza A virus strain acquires the HA segment, and possibly the NA segment as well, from an influenza virus of a different subtype. This segment reassortment can happen in cells infected with different human and animal viruses, and the resulting virus may encode completely novel antigenic proteins to which the human population has no preexisting immunity. Pandemic influenza arises when antigenic shift generates a virus to which humans are susceptible but immunologically naïve. Antigenic shift likely produced the influenza A (H1N1) virus that was the causative agent of the 1918–1919 "Spanish flu," whose lethality was unparalleled in modern times. Characterization of the reconstructed 1918 influenza virus indicated that its unique constellation of genes was responsible for its extreme virulence, with the HA, NA, and PB1 genes all contributing to its high pathogenicity. The global spread of the pandemic was almost certainly enabled by the acquisition of antigenically novel surface proteins, to which much of the world's population was immunologically naïve.

The replication cycle can be considered into 6 phases.

a. Virus Attachment
b. Virus Entry
c. Synthesis of Viral RNA
d. Synthesis of Viral Proteins
e. Packaging of RNA and Assembly of Virus
f. Virus Budding and Release

Their segmented genomes and their error-prone RNA-dependent RNA polymerases enable these viruses to undergo antigenic shift and drift, which in turn results in an evasion of the adaptive immune responses in a range of mammalian and avian species, including humans. Because of their adaptive ability, influenza viruses continue to confound efforts to produce long-lasting vaccines against the disease.

Influenza update 2018–2019: 100 years after the great pandemic[1]

Key influenza-related events since the 1918 influenza pandemic

- 1918 —Influenza A(H1N1) pandemic
- 1933—Isolation of influenza virus; development of first vaccine
- 1952—World Health Organization establishes the Global Influenza Surveillance Network
- 1957—Influenza A(H2N2) pandemic
- 1968 —Influenza A(H3N2) pandemic; antiviral drugs developed
- 2009—Influenza A(H1N1) pandemic
- 2013—First non-egg-based vaccine

- 2018—Universal vaccine studies

Influenza A and B account for almost all influenza-related outpatient visits and hospitalizations. While pigs and birds are the major reservoirs of influenza viral genetic diversity from which infection is transmitted to humans, dogs and cats have recently emerged as possible sources of novel reassortant influenza A. Obesity emerged as a risk factor for severe influenza in the 2009 pandemic. Recent studies shows that Humidity may not block transmission.

Influenza vaccine effectiveness in the 2017– 2018 influenza season was 36% overall, 67% against A (H1N1), 42% against influenza B, and 25% against A (H3N2). Studies shows that influenza serotype, with higher effectiveness in people ages 5 to 17 and ages 18 to 64 than in those age 65 and older.

A study from Japan showed that people who needed medical attention for influenza in the previous season were at lower risk of a similar event in the current season. This suggests that infection is more immunogenic than vaccination, but only against the serotype causing the infection and not the other serotypes included in the vaccine. Many studies have shown the value of influenza vaccination during pregnancy for both mothers and their infants.

Several factors including age-related frailty and iatrogenic and disease-related immunosuppression can affect vaccine effectiveness. The egg based influenza vaccine will be replace to cell-based baculovirus influenza vaccine.

All current influenza vaccines aim at the cap portion of the hemagglutinin protein. Annual antigenic drift of influenza viruses alters the cap portion of the hemagglutinin protein, requiring annual vaccine updates. The stalk portion of the hemagglutinin protein is consistent among different influenza viruses and is not altered annually like the cap portion. Thus, a vaccine aimed at the stalk portion of the hemagglutinin protein has the potential to be a universal vaccine.

My perspective on Influenza Viruses

Even though we were successful to study the biological structure and behavior of Influenza viruses, because of their antigenic drift and antigenic shift, makes it difficult to develop a vaccination which doesn’t need an yearly update, against these viruses. A vaccine aimed at the stalk portion of the hemagglutinin protein has the potential to be a universal vaccine.

6.0 Conclusion

We have studied about the Influenza A segment 1 virus from Shanghai and Ohio and compared the alignment between Global alignment and BLAST. Because of these mutation, it is difficult to make a vaccination which is very effective in reducing the flue.

7.0 References

[1] S. B. Mossad, “Influenza update 2018-2019: 100 years after the great pandemic,” *Cleve Clin J Med*, vol. 85, no. 11, pp. 861–869, Nov. 2018.

[2] N. M. Bouvier and P. Palese, “THE BIOLOGY OF INFLUENZA VIRUSES,” *Vaccine*, vol. 26, no. Suppl 4, pp. D49–D53, Sep. 2008.

[3] “How the Flu Virus Can Change: ‘Drift’ and ‘Shift’ | CDC,” 27-Sep-2017. [Online]. Available: https://www.cdc.gov/flu/about/viruses/change.htm. [Accessed: 24-Mar-2019].

[4] “Antigenic shift,” *Wikipedia*. 05-Nov-2018.

[5] "Needleman–Wunsch algorithm," *Wikipedia*. 21-Feb-2019.
[6] "Global alignment of two sequences - Needleman-Wunsch Algorithm (Theory) : Bioinformatics Virtual Lab II : Biotechnology and Biomedical Engineering : Amrita Vishwa Vidyapeetham Virtual Lab." [Online]. Available: https://vlab.amrita.edu/?sub=3&brch=274&sim=1431&cnt=1. [Accessed: 26-Mar-2019].
[7] "BLAST: Basic Local Alignment Search Tool." [Online]. Available: https://blast.ncbi.nlm.nih.gov/Blast.cgi. [Accessed: 26-Mar-2019].
[8] "Influenza A virus (A/human/Ohio/2006(H3N2)) segment 1 polymerase PB2 (PB2) gene, complete cds," Jul. 2007.
[9] "Influenza A virus (A/Shanghai/02/2013(H7N9)) segment 1 polymerase PB2 (PB2) gene, complete cds," Sep. 2013.
[10] E. Giladi, M. G. Walker, J. Z. Wang, and W. Volkmuth, "SST: An algorithm for searching sequence databases in time proportional to the logarithm of the database size.," p. 10.
[11] "ViralMultiSegProj274585 - Genome - Assembly - NCBI." [Online]. Available: https://www.ncbi.nlm.nih.gov/assembly/GCF_000928555.1#/def. [Accessed: 10-Mar-2019].
[12] "Taxonomy Browser." [Online]. Available: https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?id=1332244. [Accessed: 26-Mar-2019].
[13] "txid1332244[Organism:noexp] - Gene - NCBI." [Online]. Available: https://www.ncbi.nlm.nih.gov/gene/?term=txid1332244[Organism:noexp]. [Accessed: 26-Mar-2019].
[14] "polymerase PB2 [Influenza A virus (A/Shanghai/02/2013(H7N9))] - Protein - NCBI." [Online]. Available: https://www.ncbi.nlm.nih.gov/protein/752901089. [Accessed: 26-Mar-2019].

8.0 Appendices

Mutations Code

```
import re

"""Function to find synonymous mutation"""
def find_synonymous_mutation(seq1, seq2):

  print("********************************"
      "Mutation count from Protein Sequence (indel is counted as nonsynonymus Mutation)"
      "*********************************************")

  """ Look-up table for the nucleotide to protein dictionary"""
  genetic_code = {
    'ATA': 'I', 'ATC': 'I', 'ATT': 'I', 'ATG': 'M',
    'ACA': 'T', 'ACC': 'T', 'ACG': 'T', 'ACT': 'T',
    'AAC': 'N', 'AAT': 'N', 'AAA': 'K', 'AAG': 'K',
    'AGC': 'S', 'AGT': 'S', 'AGA': 'R', 'AGG': 'R',
    'CTA': 'L', 'CTC': 'L', 'CTG': 'L', 'CTT': 'L',
    'CCA': 'P', 'CCC': 'P', 'CCG': 'P', 'CCT': 'P',
    'CAC': 'H', 'CAT': 'H', 'CAA': 'Q', 'CAG': 'Q',
    'CGA': 'R', 'CGC': 'R', 'CGG': 'R', 'CGT': 'R',
    'GTA': 'V', 'GTC': 'V', 'GTG': 'V', 'GTT': 'V',
    'GCA': 'A', 'GCC': 'A', 'GCG': 'A', 'GCT': 'A',
    'GAC': 'D', 'GAT': 'D', 'GAA': 'E', 'GAG': 'E',
    'GGA': 'G', 'GGC': 'G', 'GGG': 'G', 'GGT': 'G',
    'TCA': 'S', 'TCC': 'S', 'TCG': 'S', 'TCT': 'S',
    'TTC': 'F', 'TTT': 'F', 'TTA': 'L', 'TTG': 'L',
    'TAC': 'Y', 'TAT': 'Y', 'TAA': '_', 'TAG': '_',
    'TGC': 'C', 'TGT': 'C', 'TGA': '_', 'TGG': 'W',
}
protein1 = ""
protein2 = ""
seq1 = seq1.upper()
seq2 = seq2.upper()
start_index_1 = seq1.find('ATG')
start_index_2 = seq2.find('ATG')
seq1 = seq1[start_index_1:]
seq2 = seq2[start_index_2:]
synonymous_mutation_count =0
non_synonymous_mutation_count = 0

for i in range(0, len(seq1),3):

    codon1 = seq1[i:i + 3]
    codon2 = seq2[i:i + 3]
    if (len(codon1) == 3) and (len(codon2) == 3):
        if (codon1.find('-') == -1) and (codon2.find('-') == -1):
            protein1 = genetic_code[codon1]
            protein2 = genetic_code[codon2]
            if str(codon1) != str(codon2):
                if str(protein1) == str(protein2):
                    synonymous_mutation_count = synonymous_mutation_count+1
                else:
                    non_synonymous_mutation_count = non_synonymous_mutation_count+1

        if(codon1.find('-') != -1) or (codon2.find('-') != -1):
            non_synonymous_mutation_count = non_synonymous_mutation_count + 1

print("synonymous_mutation_count ............. ", synonymous_mutation_count)
print("non_synonymous_mutation_count ...........", non_synonymous_mutation_count)
return
```

```
""" function to find all mutations"""


def find_all_mutation(seq1,seq2):
   mutation_count =0
   similar_count = 0
   indel_count_seq1 = seq1.count('-')
   indel_count_seq2 = seq2.count('-')
   for i, j in zip(seq1, seq2):
      if i != j:
         mutation_count = mutation_count +1
      else:
         similar_count = similar_count+1

   print("Total length        ", "........................................................", len(seq2))
   print("Same Nucleotides  count ","......................................................", similar_count)
   print("Different  Nucleotides   (mutation)  count  including  indel",  ".........................",
mutation_count)
   print("Indel count for sequence 1 ......................................................", indel_count_seq1)
   print("Indel count for sequence 2 ......................................................", indel_count_seq2)


"""To find the mutations"""


def find_mutation(sequence1, sequence2):

   find_all_mutation(sequence1,sequence2)
   find_synonymous_mutation(sequence1,sequence2)


"""  Reads the aligned file from BLAST, separates two sequences """


def read_sequence():
   infile = "BLAST-Alignment.txt"
   with open(infile, "r") as f:
      inseq = f.read()
      """ Strand=Plus/Plus is the ending of the header"""
      inseq = inseq[inseq.index('Strand=Plus/Plus'):]
      inseq = inseq[inseq.index('\n')+1:]
```

```
    inseq_list = inseq.split("\n")

    line = 1
    line2 = 3
    pattern = 'Query.*\d\s*\s'
    pattern2 = '\s*\d'
    pattern3 = 'Sbjct.*\d\s*\s'
    sequence1 = ''
    sequence2 = ''
    while line < len(inseq_list):
        str = inseq_list[line]
        match = re.sub(pattern, '',str)
        match = re.sub(pattern2, '', match)
        sequence1 = sequence1+match

        str = inseq_list[line2]
        match = re.sub(pattern3, '', str)
        match = re.sub(pattern2, '', match)
        sequence2 = sequence2 + match
        line =line + 4
        line2 = line2 + 4
    print("Sequence 1 is Shanghai and Sequence 2 is Ohio")
    return sequence1, sequence2


"""To read the nucleotides before alignment"""


def read_raw_sequence(file1,file2):
    infile = file1
    with open(infile, "r") as f:
        inseq = f.read()
        inseq = inseq[inseq.find('[gbkey=CDS]'):]
        inseq = inseq[inseq.find('\n') + 1:]
        inseq = inseq.replace('\n','')
    seq1 = inseq
    infile = file2
    with open(infile, "r") as f:
        inseq = f.read()
        inseq = inseq[inseq.find('[gbkey=CDS]'):]
        inseq = inseq[inseq.find('\n') + 1:]
        inseq = inseq.replace('\n', '')
    seq2 = inseq
    return seq1, seq2


"""Reads Protein sequence """


def read_protein_sequences():
    with open("Shanghai_H7N9_protein.txt", "r") as f:
        seq1 = f.read()
        seq1 = seq1.replace('\n','')
        seq1 = seq1.replace(' ', '')
    with open("Ohio_H3N2_protein.txt", "r") as f:
        seq2 = f.read()
        seq2 = seq2.replace('\n', '')
        seq2 = seq2.replace(' ', '')
    return seq1, seq2


if __name__ == '__main__':
    seq1, seq2 = read_sequence()
    print("************************************Statistics of two sequences after BLAST
alignment"
          "************************************")
    find_mutation(seq1,seq2)

    """ from olas alignment"""

    with open("Shanghai_Aligned1.txt", "r")  as f:
        seq1 = f.read()
        seq1 = seq1.replace('\n','')
    with open("Ohio_Aligned2.txt", "r") as f:
        seq2 = f.read()
        seq2 = seq2.replace('\n', '')
    print("----------------------------------------------------------------------------------------------------
-----")
    print("************************************Statistics of two sequences from Global
alignment"
          "************************************")
    find_mutation(seq1, seq2)


    seq1, seq2 = read_raw_sequence("Ohio_H3N2.txt", "Shanghai_H7N9.txt")
    print("-----------------------------------------------------------------------------------------------
-----")
    print("**********************************Statistics of two sequences before
alignment"
          "**********************************")
    find_mutation(seq1, seq2)
    print("-----------------------------------------------------------------------------------------------
-----")
    print("**********************************Statistics of two protein sequences from
website"
          "**********************************")
    sequence1, sequence2 = read_protein_sequences()
    find_all_mutation(sequence1, sequence2)
```

Global Alignment Code

```
import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;


public class SequenceAlignment {


        public static void main(String[] args) {



                String file1 = "Shanghai_H7N9.txt";

                String file2 = "Ohio_H3N2.txt";

                String sequenceB ="";

                String sequenceA = "";

                try
        {
                 Path path1 = Paths.get(file1);
                 sequenceA = new String ( Files.readAllBytes( path1 ) );
         Path path2 = Paths.get(file2);
         sequenceB = new String ( Files.readAllBytes( path2 ) );
        }
        catch (IOException e)
        {
          e.printStackTrace();
        }          // The penalties to apply
                int gap = -2, mismatch = -1, match = 1;
                Align(sequenceA, sequenceB, gap, match, mismatch);
          }
    public static void Align(String sequenceA,String sequenceB,int gap,int match,int mismatch) {
          int[][] scorearray = new int[sequenceA.length() + 1][sequenceB.length() + 1];
          String[][] traceBack = new String[sequenceA.length() + 1][sequenceB.length() + 1];
                traceBack[0][0] = "";
                for (int i = 1; i <= sequenceA.length(); i++)
                {
                      scorearray[i][0] = scorearray[i - 1][0] + gap;
                      traceBack[i][0] = "up";
                }
                for (int j = 1; j <= sequenceB.length(); j++)
                {
                      scorearray[0][j] = scorearray[0][j - 1] + gap;
                      traceBack[0][j] = "left";
                }
                for (int i = 1; i <= sequenceA.length(); i++) {
```

```
            for (int j = 1; j <= sequenceB.length(); j++) {
                int scoreDiag = scorearray[i - 1][j - 1] +
                        (sequenceA.charAt(i-1) == sequenceB.charAt(j-1) ?
                            match : // same symbol
                                mismatch); // different symbol
                int scoreLeft = scorearray[i][j - 1] + gap;
                int scoreUp = scorearray[i - 1][j] + gap;
                scorearray[i][j] = Math.max(Math.max(scoreDiag, scoreLeft), scoreUp);
                if (scorearray[i][j] == scoreDiag)
                    traceBack[i][j] = "diagonal";
                else if (scorearray[i][j] == scoreLeft)
                    traceBack[i][j] = "left";
                else if (scorearray[i][j] == scoreUp)
                    traceBack[i][j] = "up";
            }
        }
        TraceBack(sequenceA, sequenceB, traceBack);
        /*for (int i = 0; i <= sequenceA.length(); i++) {
            for (int j = 0; j <= sequenceB.length(); j++)
                System.out.print(scorearray[i][j] + "\t");
            System.out.println();
        }*/
        /*for (int i = 0; i <= sequenceA.length(); i++) {
            for (int j = 0; j <= sequenceB.length(); j++)
                System.out.print(traceBack[i][j] + "\t");
            System.out.println();
        }*/
    }
```

```
        public static void TraceBack(String sequenceA,String sequenceB, String [][] traceBack )
{
                int i= sequenceA.length();
                int j = sequenceB.length();
                String alignedA="";
                String alignedB="";
                int k=6;
                while (i>=0 || j>=0)
                {
                        if (traceBack[i][j].equals("diagonal"))
                        {
                                alignedA=alignedA+sequenceA.charAt(i-1);
                                alignedB=alignedB+sequenceB.charAt(j-1);
                                i--;
                                j--;
                        }
                        else if (traceBack[i][j].equals("up") )
                        {
                                alignedA=alignedA+sequenceA.charAt(i-1);
                                alignedB=alignedB+"-";
                                i--;
                                if (j == 0)
                                        j--;
                        }
                        else if (traceBack[i][j].equals("left"))
                        {
                                alignedA=alignedA+"-";
                                alignedB=alignedB+sequenceB.charAt(j-1);
                                j--;
                        if (i == 0)
                            i--;
                    }
                    if (i<0 || j<0 || (i==0 && j==0))
                        break;
                }
                try {
    BufferedWriter out = new BufferedWriter(new FileWriter("Shanghai_Aligned1.txt"));
    out.write(new StringBuilder(alignedA).reverse().toString());
    out.close();
    BufferedWriter out2 = new BufferedWriter(new FileWriter("Ohio_Aligned2.txt"));
    out2.write(new StringBuilder(alignedB).reverse().toString());
    out2.close();
}
catch (IOException e)
{
    System.out.println("Exception ");
}
                System.out.println("done ");
    }
    }
```