Hierarchical Clustering for Computational Biology

Tyler Cadigan
Ann Chou
CSC 428\589B

April 10, 2010

1 Introduction

This paper details our experiences with implementing a hierarchical clustering application. We performed such work in the context of creating phylogeny trees for sets of biological data. Hierarchical clustering is a method of grouping data points together such that elements that are more similar to one another exist within the same cluster. The membership of each cluster can provide large amounts of information depending on the problem domain. One common problem domain is for the creation of phylogeny trees from computational biology. A phylogeny tree is binary tree that charts the evolutionary distance between sets of genomic sequences. Each bifurcation in the tree represents a evolutionary action which resulted in the creation of two hosts that have similar, but distinct, genetic sequences. From a given set of modern genomic sequences from many different hosts, creation of the phylogeny tree attempts to trace back in time to determine how a particular genome sequence evolved over time between the hosts.

The process for creating a phylogeny tree is compute a distance metric between all of the modern sequences, then perform hierarchical clustering, and finally visualize the result. Each of these steps depend on the output of the previous and all come with their own challenges, algorithms and difficulties. We implemented all three steps and tested our implementation on several genomic datasets. We leverage our knowledge of computer science to tackle each of these steps due to the algorithmic nature of each step. By implementing the steps in a computer, we can get results faster and possibly more accurately than by hand. Further we can work on larger datasets on computers than we can by hand due to the speed of computation provided by modern computational resources.

The rest of the paper is as follows, section 2 more formally introduces the biological problem that we are attempting to solve and how computer science can help solve such a problem. The methodology of our approach is explained in section 3, both from a computational perspective and from a computational perspective. Analysis of our results and methodology is provided in section 4. Section 5 looks at some further applications of hierarchical clustering both in the biological and computer science research communities. Finally, section 6 provides concluding remarks and some suggestions for further work.

2 Problem

2.1 Biological Problem

The use of genomic research has changed how we trace and control epidemics. Our biological problem is to trace how encephalitis came to North America during the recent epidemic outbreak. Encephalitis is the inflammation of the brain. This potentially-fatal disease is commonly caused by viral infection. Severe cases of infection can cause permanent damage to the central nervous system and death. However, in most cases, the mild form of infection causes flu-like symptoms that lasts for 2-3 weeks. The causes of infection include Herpes virus, the equine (meaning horse), West Nile, Japanese, La Crosse, and St. Louis encephalitis viruses [6, 9, 10].

These viruses have different global distributions [10]. For example, before 1999, the West Nile virus was limited to the Old World, and Japanese encephalitis was dominant in Southeast Asia. However, in August and September 1999, New York experienced an outbreak of human encephalitis during which two people died. Meanwhile, some captive and wild birds died too. Weeks later, RNA sequencing told us that the disease was caused by a form of the West Nile virus [1].

The group of Encephalitis viruses (the equine, West Nile, Japanese, La Crosse, and St. Louis encephalitis viruses) are all transmitted by mosquitoes. The viruses grow and cycle between animal-hosts and mosquitoes [6, 9, 10, 1]. In an epidemic study, we want to know how an Old World virus traveled to the New World: if the virus was in a human, an animal such as a pet or an illegally smuggled exotic animal, or an infected mosquito that was trapped in a plane. With this information, we would be able to control the transmission of the communicable disease. We also want to know the linkage of the virus causing the new epidemic. With the linkage information, we can derive effective treatments and design effective vaccines for the virus. To derive a hypothesis of the linkage, we need to build a phylogenetic tree for the virus.

In our project, we study three bio-sequence datasets and build phylogenetic trees for each of them. The sets of data that we study are albumin gene sequences for assorted animals, mitochondrial genomic sequence for assorted animals, and genomic sequences for assorted viruses.

Albumin, a small protein produced by liver, it is the most abundant protein in blood stream. It is the servant in our blood: its presence controls the osmotic pressure of blood vessels; it transports non-water soluble particles and removes dangerous free radicals from blood. Albumin is commonly expressed in animals, including mammals, amphibians and fish. Albumin has been used as an evolutionary clock since Sarich first proposed albumin as in 1967 [15, 13]. Human albumin is of particular interest in toxicology for its ability to bind to various drugs. In our project, we run our implementation against albumin gene sequences to test if it can produce valid phylogenetic tree to fit our knowledge about the animal kingdom.

Mitochrondrion is the power house existing in cells of eukaryotes. Its most prominent roles are to produce energy and to regulate metabolism. Mitochondrial DNA is inherited as a single unit (haplotype), usually coming from the egg only. In contrast, most other nuclear genes are inherited both the sperm and the egg. Uniparental inheritance leads to little opportunity for genetic recombination between different lineages of mitochondria and makes mitochondrial DNA a useful source for studying evolutionary history of populations. Relationship between mitochondrial DNA from different individuals can be expressed as a pylogenetic tree. Mitochondrial Eve is one of the examples, whose mitochondrial DNA has been found to the ancestor of all mitochondrial DNA from people all over the world. The discovery of Mitochondrial Eve supports the hypothesis that all modern human is an expansion for African population [12]. In our project, we also run our implementation against mitochondrial DNA to see if it can produce similar result.

A virus is an enemy to our body. In contrast to another enemy, the bacteria, the virus cannot reproduce itself, but it depends on the host mechanism to do so. In the exercise, we are looking at a set of viruses that are either known to produce encephalitis like symptoms or related to the encephalitis virus. The viruses include various coronaviruses, animal-pox viruses, and dengue viruses together with the virus whose linkage we want to investigate, the Japanese encephalitis virus.

2.2 Computational Problem

To generate a phylogenetic tree the main computational problems are threefold. First, we must determine, and compute, a distance metric between every genomic sequence. Second we must perform hierarchical clustering on the given data sets, utilizing the distance metric computations. Finally, we must have some manner of visualizing the resulting phylogenetic tree. Each problem comes with its own set of difficulties that must be overcome, but all share the problem of computational efficiency. Different distance metrics have different time complexities, but none have running times faster than linear. Heirarchical clustering similarly has severl different approaches each with their own time complexity, but again running times are linear. Visualizing does not have such strict running time requirements, but rather computational complexity requirements. These requirements stem from the fact that the main focus of the project is hierarchical clustering and ensuring that aspect is correct. Thus the complexity and time afforded to working on visualization should be kept at a bare minimum.

3 Methodology

All the biological sequences data were collected from the National Center for Biotechnology Information (NBCI), available from `www.ncbi.nlm.nih.gov` by our course instructor, Marina Barsky. NCBI is a authentic and credible source of information. Originally a part of National Library for Medicine, it is US government-funded national resource for molecular biology information and an effort of international collaboration. Its genome databases are organized into six major organism groups: archaea, bacteria, eukaryotae, viruses, viroids, and plasmids. It provides access to the complete genomes of over 3,200 organisms, and provides downloads of cDNA sequences in FASTA format [11].

For the distance metric we decided to use the simple Levenshtein distance. This distance computes the number of insertions, deleteions, or substitutions of single characters between two different strings, i.e. the edit distance. The algorithm uses the dynamic programming approach to compute the minimal amount of operations needed. The algorithm first requires two strings, A and B of length n and m respectively. It operates on two arrays, $Prev$ and $Curr$ which are of length m. $Prev$ holds the previous row of the dynamic programming table, while $Curr$ holds the current row. Results are copied from $Curr$ to $Prev$ at the each pass. Each pass corresponds to finding the shortest distance between A and the current place in B. In each pass the minimum value of the cost of moving from the left adjacent, upper adjacent, or upper left adjacent, spot is computed. From the left or upper adjacent spots the cost of the move is $1 + val$ where val was the minimum cost to arrive at that spot, and the 1 corresponds to either skipping a character in A or B. From the upper left adjacent spot the cost of the move is dependent upon whether the characters at the current location in A and B match, if they do the cost is zero, otherwise it is 1 representing a substitution. The pseudocode is provided in algorithm 1. The algorithm embraces dynamic programming since at every point in a pass, computation depends on previous results. The entire edit distance table is not saved because we have no need for knowing the specific minimal alignments. We merely need the minimum number of operations. That minimum number will always be in the last entry $Prev$ once all passes have completed, since at that time every character in A will have been tested against every character in B.

The running time for computing the Levenshtein distance is order $n * m$ for strings A and B. Therefore, to compute a matrix of pairwise distances among z strings, $z * \frac{z-1}{2}$ instances of the algorithm must be run . This comes from not needing to run the algorithm for any entry on the main diagonal, any string is Levenshtein distance 0 from itself, and the matrix being symmetric. That is the Levenshtein distance between strings A and B is the same as the Levenshtein distance between strings B and A. Thus only above the main diagonal or below the main diagonal needs to be run with the algorithm, the other is just a mirror of those values.

For hierarchical clustering we used the unweighted pair group method with arithmetic mean (UPGMA) algorithm. This algorithm takes a lower triangular $n \times n$ matrix whose entries are distance measurements between the entity defining the row and the entity defining the column. These entities are termed clusters, regardless of if they have one element or many elements (as arises during the algorithm). UPGMA first creates a working copy of the distance measurements matrix, and leaves the original for reference. Then while there is still more than one cluster in the working copy, The lowest non-zero entry is located. The cluster that defines the row and the cluster that defines the column of that entry are grouped together to form a new cluster. This new cluster is added as a row and a column to the

Algorithm 1 Levenshtein distance algorithm

```
Require: String $A$ of length $n$
Require: String $B$ of length $m$
  for $i = 1$ to $m$ do
    $Prev[i] \leftarrow i$
  end for
  for $i = 1$ to $n$ do
    $Curr[0] = i$
    for $j = 1$ to $m$ do
      if $A[i] == B[j]$ then
        $Score \leftarrow 0$
      else
        $Score \leftarrow 1$
      end if
      $Curr[j] \leftarrow min(Curr[j-1]+1, Prev[j]+1, Prev[j-1]+Score)$
    end for
    $Prev \leftarrow Curr$
  end for
  return $Prev[m]$
```

working distance matrix, while the original row and column are deleted. The entries in the new row and column are computed as follows. Let the entry to be filled be in location x, y, then let cluster A be the cluster that defines row x and cluster B be the cluster that defines column y. Then the value of the entry is

$$\frac{1}{|A| * |B|} * \sum_{x \epsilon A} \sum_{y \epsilon B} dist_{orig}(x, y)$$

where $dist_{orig}(x, y)$ is the value in the original distance measurement matrix at location x, y. The values of x and y are all of the leaves in cluster A and B. The UPGMA algorithm also specifies when and how the tree is drawn. The tree is initialized with every cluster in the distance measurements matrix being a leaf. Then when the lowest entry is selected as described above, the cluster defining the row and the cluster defining the column are joined under a new root node. The total length from a leaf in one cluster to a leaf in another cluster through the new root is the found minimal value. Indicating that the length of each individual branch is half of the minimal value. These lengths can be seen on our figures in this report. The pseudocode for UPGMA is presented in algorithm 2. In the pseudocode the '$*$' operator indicates concatenating two clusters together, while the '$\times$' operator indicates standard multiplication. The running time for this algorithm is order n^2, due to the need to search the entire matrix for the smallest value.

The final computational step that was required was visualization. While the pseudocode for UPGMA does include steps on how to create the tree visualization, we decided to remove that complexity from our UPGMA implementation and instead do a post processing step

Algorithm 2 UPGMA algorithm

```
Require: lower triangular $n \times n$ matrix, $A$, of pairwise Levenshtein distances
  $cpy \leftarrow A$
  for all $c$ in $cpy$ do
    create node in tree of cluster $c$
  end for
  while $|cpy| > 1$ do
    $x \leftarrow$ lowest value in $cpy$
    $c\_clus \leftarrow cpy[0][x]$
    $r\_clus \leftarrow cpy[x][0]$
    create new cluster $(c\_clus * r\_clus)$
    connect $c\_clus$ and $r\_clus$ to new node in tree with branch lengths of $\frac{x}{2}$
    delete row containing $r\_clus$ in $cpy$
    delete column containing $c\_clus$ in $cpy$
    for all $c$ in $cpy$ do
      if $c \neq c\_clus$ && $c \neq r\_clus$ then
        $cpy[(c\_clus * r\_clus)][c] \leftarrow \frac{1}{|c\_clus*r\_clus| \times |c|} \times \sum_{x \epsilon c} \sum_{y \epsilon (c\_clus*r\_clus)} A[x][y]$
      end if
    end for
  end while
```

on our output for the visualization. We settled upon having our visualization lie in the Dot language (`http://www.graphviz.org/doc/info/lang.html`) which is part of the graphviz project. We decided upon this because of it's simple specification. All that needs to know are the nodes and the edges between them and any attributes that maybe associated with them, such as color, labels, etc. Once that is specified then it can be compiled and the output is visual representation of the graph.

The steps that were applied in the post processing to get to the visualization, were to reconstruct the tree in memory from the UPGMA output, verify tree sanity, and finally output to Dot format. None of these steps have discrete algorithms associated with them. Tree sanity refers to every non-leaf node must have an height that is greater than both of it's children. The height is defined as the value of the entry that determined that two clusters should be merged in the UPGMA implementation. This value was recorded and printed as part of the output from UPGMA implementation along with the clusters. To produce Dot format output first a depth first traversal is performed on the tree to reach all nodes and their edges. Then a level order traversal is performed to group all of nodes with the same logical height together in the visualization. Logical height refers to the standard theoretical computer science notion of height, the height of a node x is the length of the longest path in the subtree rooted at x.

4 Analysis

4.1 Biological Perspective

Our albumin result validates the correctness of our implementation, with one interesting finding. The interesting finding comes from the evolution phenomenon of orthologs and paralogs. Orthologs and paralogs are two kinds of homologs, homogeneous sequences descended from a common ancestral sequence. Orthologs are genes in different species that evolved from a common ancestral gene by speciation. Orthologs do not necessary have same functions. In contrast, paralogs are genes within a single species related by duplication within a genome [4]. As seen in fig. 1, cow, mice and frog results form clusters of their own. The cows are more closely related to another mammal, mice. From the common ancestor albumin gene, it has speciated into the common ancestor for the frog and salmon and the common ancestor for the cow and the mouse.

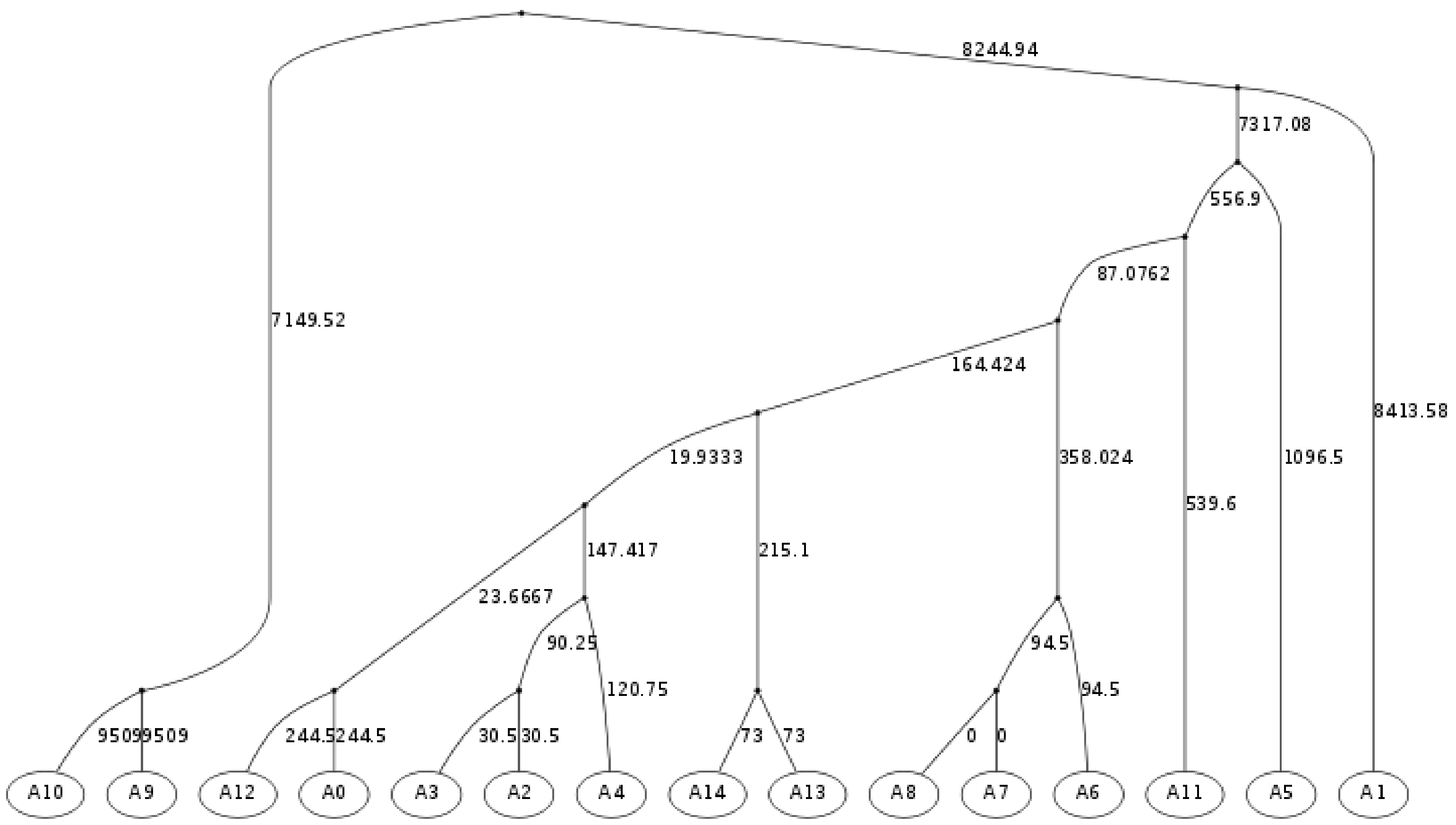

Figure 1: Albumin Phologenetic Tree

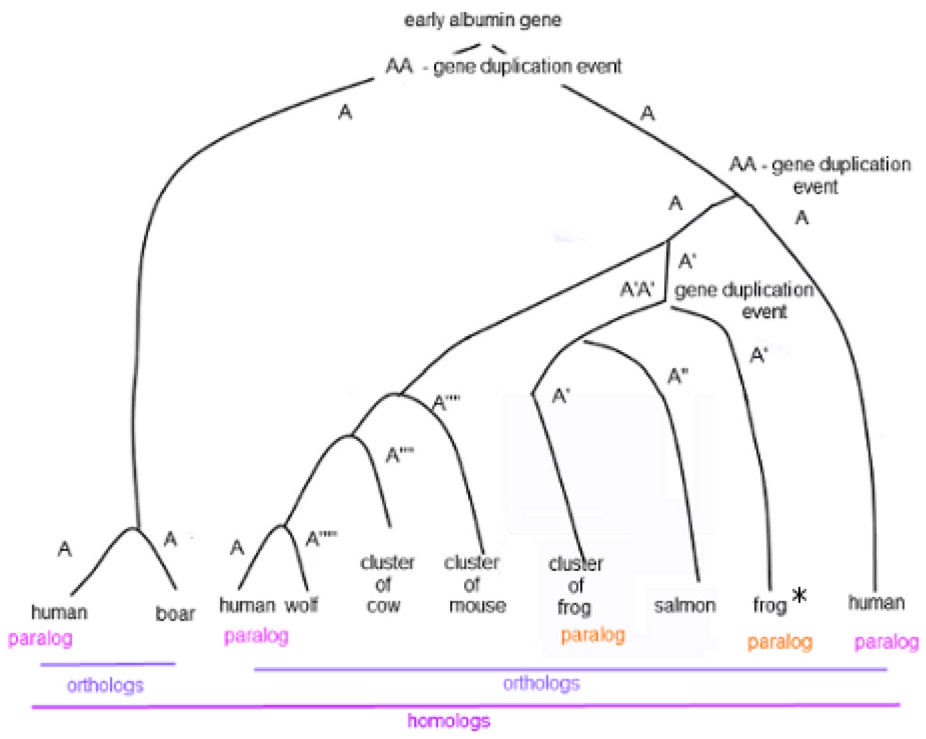

Figure 2: Orthologs and Paralogs of Albumin Results

The observation of human albumin genes appears at very far left and very far right of the whole tree and the left of the left subtree can be explained by paralogy. Paralogy describes homologous genes within a single species that diverged by gene duplication. As shown by a simplified diagram of fig. 2, at least two gene duplication events took place in the evolution history of human albumin gene: one in the common ancestor of human, boar, frog, salmon, mouse, cow and wolf, another one in the common ancestor of human, frog, salmon, mouse, cow and wolf. In addition, gene duplication event took place in the common ancestor of frog and salmon as well. Indeed, studies on albumin phylogeny and mitochondrial DNA phylogeny on African clawed frogs (Xenopus) has shown the complete genome duplications took place at least six times. African clawed frogs, Xenopus laevis, was known to have double chromosome number and genome size of its congener Xenopus (formerly Silurana) tropicalis [8, 2, 5].

Our mitochondrial result further validate the correctness of our implementation. Our result is consistent with the hypothesis of Mitochondrial Eve. As seen in fig. 3, all the human results are clustered together, with the African halogroups closer to each other and the Neolithic Greece halogroup (6800 to 3200 BC) branching off. The overall tree is also consistent our current understanding of evolution: from the common ancestor, it has speciated into the ancestor for fish and the ancestor for mammals. As seen in the fig. 3, mammals such as mice, hippopotamus, giraffe, camel and blackbuck are clustered on the left hand side, the fish such as puffer fish, pollock, haddock and cod are clustered on the right hand side.

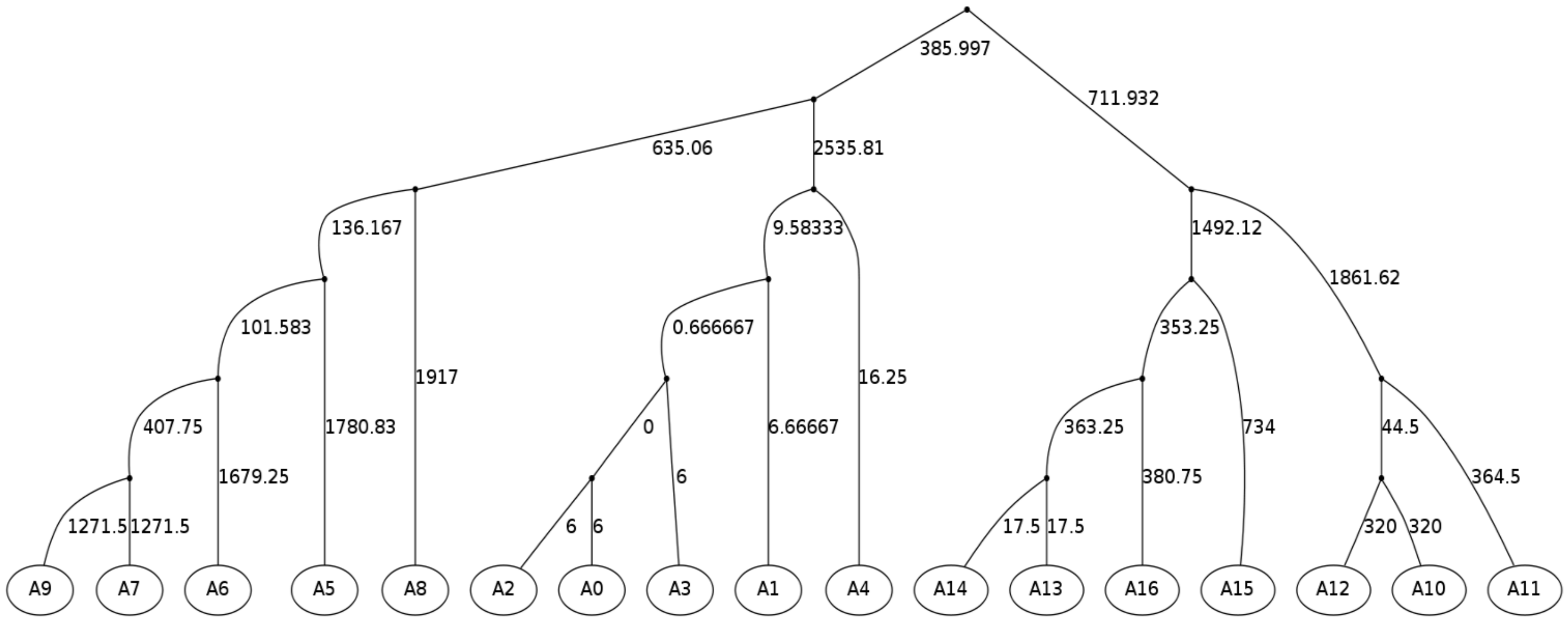

Figure 3: Mitochondria Phologenetic Tree

With the validation tests with the mitochondrial genome set and albumin gene set, we have confidence that our pylogenetic tree for virus is correct. Figure 4 shows three major clusters, the cluster of animal pox, the cluster of human-host coronavirus, and the cluster of dengue virus, cow-host coronavirus and Japanese encephalitis virus. From the large distance ($\approx$24000 basepairs) between first speciation point to animal pox and first speciation point to dengue virus and coronavirus, we can conclude that animal pox, which is spread by rodents, is very distant to coronavirus, dengue virus and Japanese Encephalitis. And coronavirus, dengue virus and Japanese Encephalitis are much closer to each other than to animal pox.

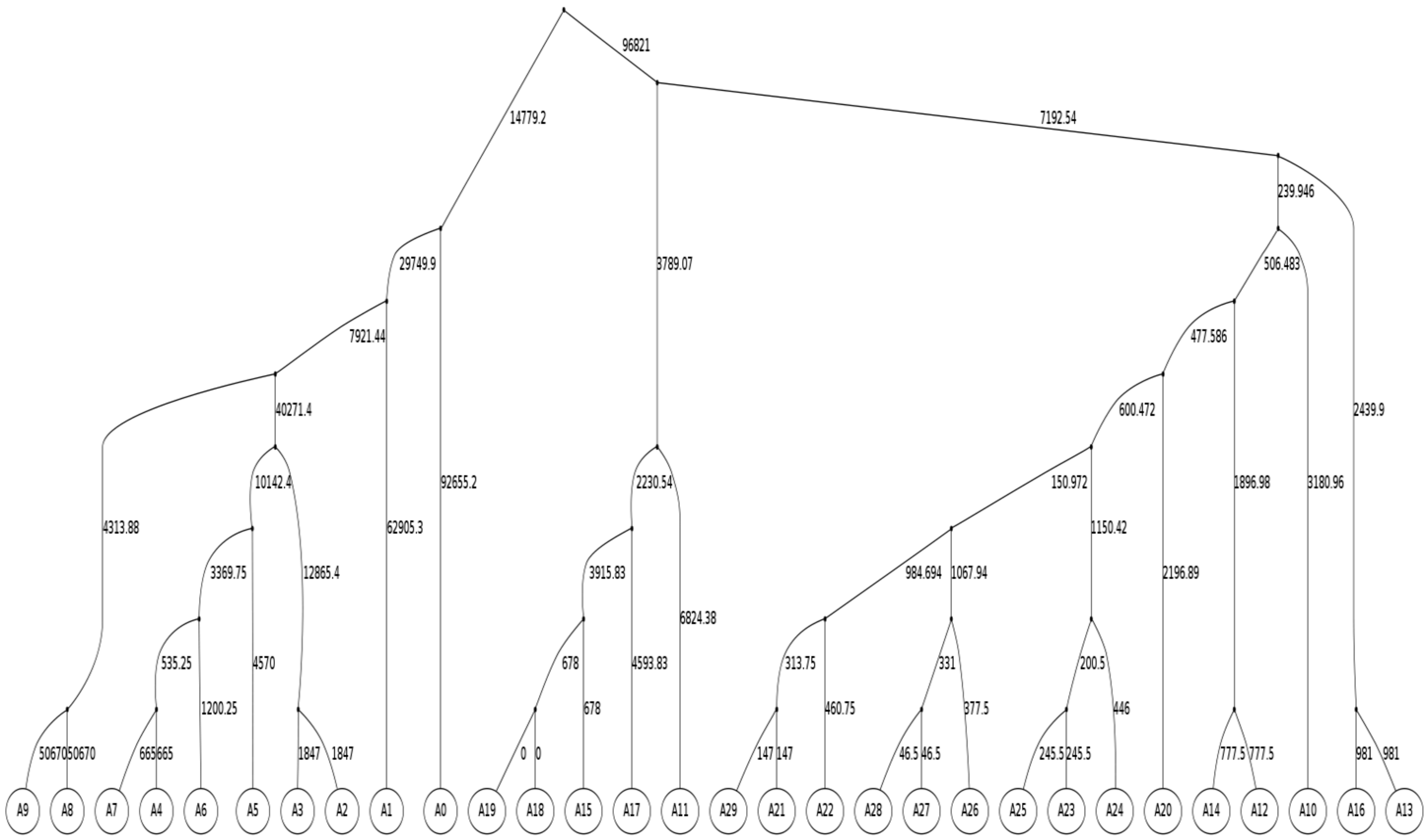

Figure 4: Virus Phologenetic Tree

Within the subtree for animal pox, we can derive the order of appearance are that of bird-host (Canarypox, Fowlpox), camel-host (Camelpox CMS, M-96), monkey-host (Monkeypox Congo 2003, Liberia 1970, COP-58, Sierra Leone) , and lastly insect-host (AmEPV, MsEPV). Presumed West African Monkeypox COP-58 is of the closest linkage with the Monkeypox Sierra Leone in 1970-80s and next closest to Monkeypos Liberia 1970. Monkeypox Congo 2003 is of another linkage.

The right subtree consists of coronavirus, dengue virus and Japanese Encephalitis, we can derive that the coronaviruses evolved first from their common ancestor, then Japanese Encephalitis and finally dengue viruses. Under the subtree for the dengue viruses, we can see the three serotypes are clustered separately: the left cluster is Dengue type 1 (DENV1 2007 Vietam, DENV1 2006 Vietam, DENV1 1997 Brail), the middle one is Dengue type 3 (DENV3 2008 Vietam, DENV3 2007 Vietam, DENV3 1998 Indonesia) and the right one is Dengue type 2 (DENV2 Vietam, DENV2 2002 Nicargua, DENV2 2002 Taiwan).

The result of coronavirus is a little more difficult to interpret. There are huge genetic differences within the species of coronavirus. As seen in fig. 4, coronavirus forms two major clusters, one appears on the right and another on the left of dengue virus. Our first impression is that there is gene duplication event taken place in coronavirus or its ancestor. However, when we slip the subtree of animal-host coronavirus and dengue virus horizontally, this observation disappears. Under close examination, we found that bovine coronavirus appears in both left and right coronavirus trees. Bovine coronavirus is the first outgroup in both trees as well. Therefore in both left linkage which is predominant by human-host coronavirus and the right linkage which is predominant by animal-host, bovine coronavirus is the common ancestor. Bovine CoV Quebec is more closely related to human coronavirus but Bovine CoV KCD1 is more closely related to the animal-host coronavirus.

Within the left subtree that is pre-dominant with human coronavirus, Bovine CoV Quebec is of closest distance to CoV OC43 1967. Then Human CoV HKU1 is next closest to it. Bat-host SARS CoV 2004, which causes the SARS pandemic, is a early branch-off to this group, which is classified as group 2 coronavirus in the literature [7]. Within the right subtree that is pre-dominantly animal-host coronavirus, Canine CoV CB/05 and Canine CoV INSAVC-1 are closely related each other, and so are bat-host CoV 2006 Shandong and Bovine CoV KCD1. CoV 2006 Guangxi, which host is Asian leopard cat, also falls within this linkage, which is classified as group 1 coronavirus in the literature [7].

The virus under investigation, Japanese encephalitis virus, is sandwiched between the dengue virus and the animal-host coronavirus. The absolute distance suggests that Japanese Encephalitis virus is more closely related to dengue virus than to the coronavirus. The phylogenetic tree structure suggests that coronavirus is its ancestor. In other words, Japanese Encephalitis is a early branch off of dengue virus.

4.2 Computational Perspective

Each of the three computational challenges were successfully met and over come. Though each did have their own share of difficulties and from them lessons to learn. Finally all of them do have some implications to other computational problems and domains.

For Levenshtein distance the single most crucial part of the entire algorithm, in relation to the running time, is in copying $Curr$ to $Prev$. Our initial implementation did this by naively copying each element from $Curr$ to $Prev$. This increases the running time by a power of two, since m elements must be copied n times. For a large number of (short) sequences, or for a small number of long sequences, this extra running time becomes significant. Since some Mitochondrian sequences were over $200,000$ characters in length generating the distance measurement matrix was taking multiple days. Once we realized that this was the bottleneck we were ale to overcome it by low level memory management provided by our implementation language. The language provided direct copying of arbitrary amounts of memory. Since arrays are stored as contiguous blocks of memory it allowed us to transfer the entire array in one operation. This reduces the additional complexity to a constant order, greatly increasing the throughput of the implementation. But this was not enough to get the implementation to run in a reasonable amount of time. It was still taking around a day to compute the matrix fro all pairs. To overcome this we increased the compiler optimization. While this does allow for the possibility of incorrect results to crop up due to over aggressive optimization, we extensively tested correct operation of our algorithm and trust in the developers of gcc. After this we were able to gather the distance matrix against all pairs in about 10 hours. This ended up being the bottleneck over the entire project, more time was spent in this phase than all the others combined.

For UPGMA, we feared that the insertion of a new row and the deletion of a row and column in the working matrix would be our greatest problem. But in reality it did not end up being the one that gave us the most grief. Due to the language of our implementation having heavy support for arrays (boht single and multi-dimensional), these operations were trivial. What did cause us the most grief was the type weakness of the language. Our working matrix ended up being strings instead of numbers, so when we applied numerical operations to them, they would not return the correct result. But in the applying of these operations, no warning or error or any other sort of notification alerted us as to a possible problem. The language would continue to execute and implicitly convert the operation such that it would work on with the operands. Only after significant amounts of debugging were we able to identify and remedy this problem. Once fixed our implementation worked flawlessly.

For visualization the most challenging part was the actual output into Dot format. The reconstruction of the clustering tree in memory was straight forward due to it being a complete binary tree. Likewise the sanity checking was easily performed recursively on the binary tree. The output is where things became complicated. When compiling a Dot file the compiler doesn't produce output that has distinct levels from the leaves to the root. Rather it sequentially reads from the file and creates a new level in the tree for every new node that it encounters. This was a problem due to our generating the Dot file by performing a depth first traversal of the tree. Thus in a diagram that should have all leaves along the bottom and the root at the top, the actual display was an extremely tall tree that had as many levels as there were leaves. In order to correct this we had to perform a second traversal, that proceeded through the tree in level order, such that we could output what nodes should be on the same level as one another. The combination of these things allowed us to output the complete phylogenetic trees that are seen in this report. The primary benefit, which turned into a limitation, of using the Dot format is that we were able to just compile it and get an appropriate visualization. We have very little control over how the edges are drawn or the placement of the nodes, both are under the control of the compiler. This as can be seen from the figures in the report, where most of the edges have distinct curves to them. This makes the trees have an appearance of being balanced but in fact this is not the case. Looking at the lengths of the edges it quickly becomes clear parts of the tree are far from proportional. This does have a strong implication in the interpretation of the UPGMA results, it means that the initial distance measurement matrices were not ultrametric. The ultrametric property states that for any three entries in the distance measurement matrix, M_{ij},M_{ik}, M_{jk}, two are the same value and the third is of equal or lesser value. If this property is satisfied then it means that the evolutionary rate for every organism in the matrix is the same and from every root both branches will have the same length. Thus with our phylogenetic trees indicating that that our distance matrices do not have the ultrametric property, then we can conclude that organisms we are investigating have different rates of evolution.

The overall implications that we can draw from our implementations is that even though the algorithms run with acceptable complexities, on large inputs they are still quite slow. Often this aspect is missed in other projects because the input sizes are kept appropriately small such that naive implementations can still run fast. In our project there was little room for missteps since any additional complexity would have great impact on the project as a whole.

5 Further Applications

Similar applications have been used in real-life for investigating and monitoring the West Nile virus epidemic, 2003 severe acute respirator syndrome (SARS) pandemic in Asia, influenza epidemic in China and other epidemics. Epidemiology of the West Nile virus has been closely monitored in Europe. In Spain, researchers have performed characterization two isolates obtained from two golden eagles. Their complete genome sequence comparisons revealed high identity between these isolates and close relationship with other Western Mediterranean WNV strains isolated since 1996. Phylogenetic analysis within this group indicated that two distinct phylogenetic groups have emerged from earlier strains [14].

Virological surveillance in southern China was taken as an aftermath after the 2003 SARS. A group of Hong Kong researchers have performed phylogenetic analyses on this virological surveillance data. They have discovered that surveillance novel SARS coronaviruses detected from wild Asian leopard cats and Chinese ferret badgers fell into an outgroup phylogenetic relationship with respect to other coronaviruses and had low amino acid similarity to all known coronavirus groups. The result indicates that the novel SARS virus diverged early in the evolutionary history of coronaviruses and suggests a previous undiscovered evolution pathway [3].

A group of researchers in China have employed similar approach to understand an influenza virus circulated from 2001 to 2006 in Chinas Liaoning local area. They has extracted, transcribed and sequenced the viral RNA. Then the researcher draws the phylogenetic trees according to deduced amino acid sequences of influenza virus H3N2 from 2000 to 2006 in the NCBI database. The phylogenetic tree shows Liaoning H3N2 2006 strains and Zhejiang 2005 strains are similar to WHO Northern hemisphere winter 2006-2007 Vaccine A/Wisconsin/67/2005 (H3N2)-like virus and grouped together to form an independent cluster [16].

Hierarchical clustering has also been used in other areas of computer science. One specific application is for document comparison and version control. When there is a large repository of documents and many users can access and modify those documents, it becomes crucial to understand how the have arrived in their current state. In this way accountability and security of the document can be ensured. In principle to find the ”evolution” of a document the same process that we went through would be applied to it. Instead of DNA sequences from multiple species, it would be multiple revisions of the document. Then clustering and visualization would be performed.

The theoretical work on clustering has not finished yet either. The major realm where this work is being done is in the context of data streaming. Data streaming is where there is a large amount of data flowing though a given point. That point has limited memory and time to investigate the data. Furthermore it only gets to see each piece of data exactly once. Further at the point statistical information about the data is to be gathered. Such statistical information can include the types of data that have gone through the point. To accomplish this variants of the methodology presented in this report are used and are being currently developed.

6 Conclusions

In conclusion, we have overcome challenges and successfully solved our three-stage computational challenges for building phylogenetic trees for three given sets of biological sequences. For the whole project, we have implemented 100 lines of codes for Edit distance (C++), 300 lines of codes for UPGMA(Python), 200 lines of codes for visualization (C++) and 100 lines of codes for supporting problem (C++). We have countered problems in each of the stages: Timely computing for constructing distance matrix, hidden data typing errors until run time in updating distance matrix for UPGMA, and resulting phylogeny being visualized as a long tree with leaves at every level because of default depth-first traversal. The first problem, efficiency problems for distance matrix construction, was solved by low memory management and compiler optimization. With much debugging, we discovered the cause of the second problem and were able to solve it. We solved the third problem by forcing a second level-order traversal of the phylogenetic tree.

In total, we have run our implementation against three datasets, albumin gene sequences for assorted animals, mitochondrial genomic sequences for assorted animals and genomic sequences for animal pox, coronavirus, Japanese Encephalitis virus and dengue virus. The running time for each of data set is acceptable. We used the first two sets, albumin and mitochondrion, as the validation sets for our implementation. Both albumin and mitochondrion are referred to as evolutionary clocks. We are confident that our implementation is correct because the results for both sets are consistent with the current knowledge of evolution. In addition, from the pylogenetic tree based on albumin, we also observe gene duplication events taken place in African clawed frog and its ancestors; these events have been studied by numerous researchers. Finally, the run against our test set, viruses, has shown three distinct clusters, with the cluster of animal pox very distant from the cluster of human-host coronavirus and from the cluster of dengue virus. We interpret that coronavirus, dengue virus and Japanese Encephalitis are much closer to each other than to animal pox.

Though our implementation was successful on the data sets we were provided with we feel that there is still further work to be done. First and foremost would be to streamline the process of going from data set to phylogeny tree. We currently have to go through 19 steps, with a significant amount of manual intervention needed. While these steps were manageable for the small number of data sets that we worked with, it does not scale well. By reducing the number of steps, or at least automating the transition between steps, the process would become more "fire-and-forget" which would allow us focus on other important tasks. the other major part of future work would be to run our implementation on larger amounts of data. The data sets were quite sparse, only 15 to 30 sequences in each set, thus our conclusions are quite general. By running on more sequences in a set we would have a much clearer picture of the what happened during evolution and would thus be able to draw much more in depth conclusions. By running on other sets entirely we would be able to both simply draw more conclusions and to better correlate the results with one another. As it stands our data sets are very disjoint from one another, both in biologic function and the genetic hosts.

We feel that the project has been a worthwhile exercise, since it has had strong components both in biology and in computer science. Even though there were many points of confusion during the project about expectations and applicability of provided data sets, we believe that we adequately resolved these issues and are able to present our results with confidence. We further feel different backgrounds, one was strong in biology and the other strong in computer science, has been beneficial rather than a hindrance. By having backgrounds in the two fields we were able to translate the biological problem to a computer science problem, determine an implementation of the problem, then interpret the results back into a biological context. We believe that this process of taking problems and converting them to different domains and then being able to reconvert back crucial to both computer science and biology as well as other fields. The main difficulty comes in how effectively communicate the problems between domains such that domain experts are able to understand the problem and solve them. This course has provided us with some insight into numerous problems in both biology and computer science that can be effectively solved using knowledge from one another. We hope to be able to keep this in mind as we proceed forward with our careers so that future difficult problems can also be solved.

References

[1] John F. Anderson, Theodore G. Andreadis, Charles R. Vossbrinck, Shirley Tirrell, Edward M. Wakem, Richard A. French, Antonio E. Garmendia, and Herbert J. Van Kruiningen. Isolation of West Nile Virus from Mosquitoes, Crows, and a Cooper's Hawk in Connecticut. *Science*, 286(5448):2331–2333, 1999.

[2] CA Bisbee, MA Baker, AC Wilson, I Haji-Azimi, and M Fischberg. Albumin phylogeny for clawed frogs (Xenopus). *Science*, 195(4280):785–787, 1977.

[3] B. Q. Dong, W. Liu, X. H. Fan, D. Vijaykrishna, X. C. Tang, F. Gao, L. F. Li, G. J. Li, J. X. Zhang, L. Q. Yang, L. L. M. Poon, S. Y. Zhang, J. S. M. Peiris, G. J. D. Smith, H. Chen, and Y. Guan. Detection of a Novel and Highly Divergent Coronavirus from Asian Leopard Cats and Chinese Ferret Badgers in Southern China. *J. Virol.*, 81(13):6920–6926, 2007.

[4] NCBI Education. NCBI Education. `http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/Orthology.html`, February 2009.

[5] Ben J. Evans, Darcy B. Kelley, Richard C. Tinsley, Don J. Melnick, and David C. Cannatella. A mitochondrial dna phylogeny of african clawed frogs: phylogeography and implications for polyploid evolution. *Molecular Phylogenetics and Evolution*, 33(1):197–213, 2004.

[6] Debapriya Ghosh and Anirban Basu. Japanese encephalitis-a pathological and clinical perspective. *PLoS Negl Trop Dis*, 3(9):e437, 09 2009.

[7] Alexander E. Gorbalenya, Eric J. Snijder, and Willy J. M. Spaan. Severe Acute Respiratory Syndrome Coronavirus Phylogeny: toward Consensus. *J. Virol.*, 78(15):7863–7866, 2004.

[8] Uffe Hellsten, Mustafa Khokha, Timothy Grammer, Richard Harland, Paul Richardson, and Daniel Rokhsar. Accelerated gene evolution and subfunctionalization in the pseudotetraploid frog xenopus laevis. *BMC Biology*, 5(1):31, 2007.

[9] R. S. Lanciotti, J. T. Roehrig, V. Deubel, J. Smith, M. Parker, K. Steele, B. Crise, K. E. Volpe, M. B. Crabtree, J. H. Scherret, R. A. Hall, J. S. MacKenzie, C. B. Cropp, B. Panigrahy, E. Ostlund, B. Schmitt, M. Malkinson, C. Banet, J. Weissman, N. Komar, H. M. Savage, W. Stone, T. McNamara, and D. J. Gubler. Origin of the West Nile Virus Responsible for an Outbreak of Encephalitis in the Northeastern United States. *Science*, 286(5448):2333–2337, 1999.

[10] Tod Mundy. Encephalitis. `http://www.emedicinehealth.com/encephalitis/article_em.htm`, October 2005.

[11] NCBI. Welcome to NCBI. `http://www.ncbi.nlm.nih.gov/`, October 2009.

[12] UC Museum of Paleontology. Genealogy enthusiasts mine dna for clues to evolutionary history. `http://evolution.berkeley.edu/evolibrary/news/071101_genealogy`, November 2007.

[13] Vincent M. Sarich. Generation time and albumin evolution. *Biochemical Genetics*, 7(3):205–212, Dec 1972.

[14] Elena Sotelo, Jovita Fernandez-Pinero, Francisco Llorente, Montserrat Aguero, Ursula Hoefle, Juan M Blanco, and Miguel A Jimenez-Clavero. Characterization of West Nile virus isolates from Spain: new insights into the distinct West Nile virus eco-epidemiology in the Western Mediterranean. *J. Virol.*, 395(2):289–97, 2009.

[15] Donald G. Wallace, Linda R. Maxson, and Allan C. Wilson. Albumin Evolution in Frogs: A Test of the Evolutionary Clock Hypothesis. *Proceedings of the National Academy of Sciences of the United States of America*, 68(12):3127–3129, 1971.

[16] Shao-Hui Wu, Yue-Long Shu, Zhuo Zhao, Wen-Qing Yao, Wei Yu, Mei-Mei Zhang, Jian-Qiu Cui, Min Liu, Rong-Hua Fu, and Xiao-Guang Zhao. An analysis on genetic characterization of HA1 gene of influenza virus subtype H3N2 circulated from 2001 to 2006 in Liaoning local area. *Zhonghua Shi Yan He Lin Chuang Bing Du Xue Za Zhi*, 23(3):174–6, 2009.

7 Appendices

Table 1: Figure 1 Label to Sequence Information

Labels	Name used in report	Common name	Identifier
A0	human	human	gb:AF542069.1
A1	human	human	gb:M12523.1
A2	cow	European taurine cattle	gb:AF542068.1
A3	cow	European taurine cattle	emb:Y17769.1
A4	cow	European taurine cattle	emb:X58989.1
A5	frog*	African clawed frog*	ref:NM_001004887.1
A6	frog	African clawed frog	gb:M21442.1
A7	frog	African clawed frog	gb:M18350.1
A8	frog	African clawed frog	ref:NM_001087775.1
A9	boar	wild boar	gb:AY663543.1
A10	human	human	gb:M12523.1
A11	salmon	Atlantic salmon	ref:NM_001123692.1
A12	wolf	gray wolf	dbj:AB090854.1
A13	mouse	house mouse	emb:AJ457860.1
A14	mouse	house mouse	emb:AJ011413.1

Table 2: Figure 3 Label to Sequence Information

Labels	Name used in report	Common name	Identifier
A0	human haplogroup H2a1 E. Arfrica	human haplogroup H2a1 E. Africa	gb:FJ800808.1
A1	human haplogroup H3c sub Sarharan Africa	human haplogroup H3c sub Sarharan Africa	gb:FJ794693.1
A2	human haplogroup H1c1 W. & C. sub-Saharan Africa	human haplogroup H1c1 W. & C. sub-Saharan Africa	gb:FJ798928.1
A3	human haplogroup H5 Africa	human haplogroup H5 Africa	gb:FJ794473.1
A4	human haplogroup J2b Neolithic Greece	human haplogroup J2b Neolithic Greece	gb:FJ445408.2
A5	Mouse	House mouse	dbj:AP003428.1
A6	Hippopotamus	Hippopotamus	dbj:AP003425.1
A7	Giraffe	Giraffe	dbj:AP003424.1
A8	Bactrian Camel	Bactrian Camel	dbj:AP003423.1
A9	Blackbuck	Blackbuck	dbj—AP003422.1
A10	Fugu Puffer fish	Fugu Puffer fish	dbj:AP009536.1
A11	Oblong blow fish	Oblong blow fish	dbj:AP009535.1
A12	Eyespot Puffer	Eyespot Puffer	dbj:AP009534.1
A13	Norwegian pollock	Norwegian pollock	emb:AM489719.1
A14	Norwegian pollock	Norwegian pollock	emb:AM489718.1
A15	Haddock	Haddock	emb:AM489717.1
A16	Atlantic cod	Atlantic cod	emb:AM489716.1

Table 3: Figure 4 Label to Sequence Information

Labels	Name used in report	Common host(s)	Transmission vector	Geographic Location	Identifier
A0	Canarypox	bird	insect		Canarypox virus ATCC VR-111
A1	Fowlpox	poultry			Fowlpox virus HP1-438 Munich
A2	Camelpox CMS	camel	rodent		gb:AY009089
A3	Camelpox M-96	camel	rodent		gb:NC_003391
A4	MPV COP-58	monkey	rodent	W Africa	gb:AY753185
A5	MPV Congo 2003	human	rodent	Congo	gb:DQ011154
A6	MPV Liberia 1970	human	rodent	Liberia	gb:DQ011156
A7	MPV Sierra Leone	monkey	rodent	Sierra Leone	gb:AY741551
A8	AmEPV	insect			gb:NC_002520
A9	MsEPV	grasshopper, locust			gb:NC_001993
A10	CoV 2006 Guangxi	Asian leopard cat		Guangxi, China	gb:EF584908
A11	SARS CoV 2004	Bat		China	gb:DQ648856
A12	Canine CoV CB/05	Dog			gb:DQ112226
A13	CoV 2006 Shandong	Bat		Shandong, China	gb:EF434381
A14	Canine CoV INSAVC-1	Dog			gb:D13096
A15	Bovine CoV Quebec	Cattle		Quebec	gb:AF220295
A16	Bovine CoV KCD1	Cattle		South Korea	gb:DQ389632
A17	Human CoV HKU1 N24	human		Hong Kong	gb:DQ415901
A18	Human CoV OC43 1967	human			gb:AY585228
A19	Human CoV OC43 1967	human			gb:AY585228
A20	Japanese encephalitis	human	mosquito		gb:AY849939
A21	DENV1 1523/2007 Vietnam	human	mosquito	Vietnam: South	gb:EU677151
A22	DENV1 1997 Brail	human	mosquito	Brazil	gb:AF311957
A23	DENV2 2002 Nicargua	human	mosquito	Nicaragua: Managua	gb:EU482634
A24	DENV2 2002 Taiwan	human	mosquito	Taiwan	gb:DQ645547
A25	DENV2 Vietnam	human	mosquito	Vietnam: South	gb:FM210210

Continued on Next Page…

Table 3 – Continued

Labels	Name used in report	Common host(s)	Transmission vector	Geographic Location	Identifier
A26	DENV3 1998 Indonesia	human	mosquito	Indonesia: Sumatra	gb:AB189128
A27	DENV3 2008 Vietnam	human	mosquito	Vietnam: South	gb:FJ461334
A28	DENV3 2007 Vietnam	human	mosquito	Vietnam: South	gb:FJ562097
A29	DENV1 2006 Vietnam	human	mosquito	Vietnam: south	gb:EU482817