Group 11

Cyrus Liang

Patrick Solis

Prof. Heller

CS123A - Sec 01

November 17, 2016

BReast CAncer Genes Compared Across Species

**Background:**

**Problem:**

We want to find out more about how and why the breast and ovarian cancer susceptibility gene (BRCA1) works. We will be comparing the human BRCA1 gene to other mammals to see what factors lead to the gene being or not being expressed. Our project is mainly to test how feasible it is to recreate the methods used by bioinformatics scientists in programming multiple sequence alignments and creating guide trees.

**Importance:**

It is important for us to find out more about breast cancer since it affects a large number of Americans. From breastcancer.org, women in America have an average 1 in 8 chance of developing breast cancer if their life expectancy is 80 years. Also, the Susan G Komen website states that “In the U.S., between 1 in 400 and 1 in 800 people have a BRCA1/2 mutation.” If we compare the human gene to mammals, we can find out more about the functions of specific parts of the genes and use that to better target mutations.

**Data Type:**

Our data collected is from the lab we did in our bioinformatics course. It consists of modern human, bornean orangutan, house mouse, western gorilla, gray wolf, cattle, and the modern human mutation variant of the BRCA1 gene. We sampled our data from the National Center for Biotechnology Information (NCBI) database as amino acid (AA) strings. We used AA strings because we wanted to look at the biological functions of the BRCA1 gene and not necessarily the homology of the species we were comparing. The strings were stored in .txt files as lines of characters. This data gets parsed through and mutated into a difference table based on the BLOcks SUbstitution Matrix (BLOSUM). From there, the table is translated into a Weighted Pair Group Method with Arithmetic Mean (WPGMA) guide tree.

**Type of Analysis:**

For our project, we are calculating a difference table by using simple local alignment scores. Afterwards, the data is going to be passed through a tree building algorithm, WPGMA. We are going to look at the guide tree we created and compare that to the trees we find in peer reviewed articles. We are also comparing our method with the big league scientists and seeing what they did to get better results. By doing this, we get to find out smarter ways to process data, and we get to learn what types of data are better for the hypothesis we are aiming for.

**Expected Results:**

Because of the methods we are doing, our expected outcome is going to be slightly off from the prefered outcome. The prefered methods are a bit beyond our capabilities without us simply borrowing someone else’s program to run the data. Implementing algorithms we knew we had the time and resources to do means that our results might be imprecise.

**Method:**

**Data:**

The AA strings we got were from lab 3 “BRCA1: Good gene, bad name.” The data is from NCBI’s database. We were given the data in our lab 3 download package.

**Procedure:**

In "Human, canine, and murine BRCA1 genes: sequence comparison among species," the scientists compared sequences across different species by comparing sequence alignments. "Identification of conserved gene expression features between murine mammary carcinoma models and human breast tumors" went further and compared protein sequences in order to find biologically important patterns in mouse breast cancers. We based our project on the lab "BRCA1: Good gene, bad name."

We did our best to recreate a multiple sequence alignment in Java. Our output is a .txt file that is a difference chart. From there, we have another program that takes the .txt file and turns it into a WPGMA guide tree. Both programs were written in Java. We used Netbeans to code the pairwise alignment, and Eclipse to code the guide tree. GitHub was used for version control, and Google Drive was used to share files. For communication, we mostly used text message, but we used Slack for more important links that needed to stay visible for the duration of the project. We decided to use Java because both of us were familiar with Java and were programming on different operating systems.

**Sequence Alignment:**

The Pairwise Alignment Java program is split into three classes: Main, Global Alignment, and Grid. This program takes multiple sequences and calculates all the differences between each score. Main is 146 lines long, reads an input file containing file paths to the sequences to align, and writes an output file for the WPGMA tree program to read. Main creates Global Alignment objects to call the “getBestScore()” method to calculate scores. Global Alignment is 153 lines long. When objects are created in Global Alignment, a Grid object is created, too. Grid is 383 lines long. Grid automatically populates a 2D array holding Needleman-Wunsch values. The scoring guidelines we used were, if there’s a gap, that means -2 points. Mismatch or match scores were based on the BLOSUM62 Substitution Matrix. Gaps are penalized because they are insertions and deletions where the reading frame is changed, and this is a drastic metamorphosis that we do not expect organisms to survive.

Main uses java.io.\* and java.util.ArrayList libraries. When running the program, it takes 2 arguments from cmd line. These are the path to input file (Figure 1) and output file. Using FileReader and BufferedReader, we go through each line in the .txt file and store the paths to the sequences in a String ArrayList because ArrayLists do not have predetermined lengths. From there, the size of the Array that holds the file paths is created because it is easier to access indexes in Arrays. The String Array holding file paths is called “filenames.” After file paths are stored for easy access, Main starts to create Global Alignment objects. We go through Array “filenames” and create handshakes between sequences. This makes it so that each sequence is compared to the other sequences once. The best score from each grid generated by the Needleman-Wunsch algorithm is stored in another String ArrayList called “pairScores.” Once again, the ArrayList is transferred into a String Array “multiSeqAlignment” for easier access. Finally, we take all the calculated data from Main and write it into an output .txt file (Figure 2).

Global alignment is straightforward. It takes the file paths to sequences that are passed to it and reads the amino acid sequences. The program just concatenates the lines into Strings and stores them in an Array of Strings called “aaA” and “aaB”. A Grid object is created in the constructor. There are basic methods that allow for reading and morphing private variables. The global score is just the cell where the last row and column intersect. The best score is the highest value in the entire 2D grid. This value is found by simply running a nested for-loop and comparing the values to each other, only keeping the greatest integer. We also have a line that calls and stores the final global score from the grid, but we don’t use that value because it is not accurate compared to the local highest score.

Grid is where the algorithm is ran, so it is more complicated than the rest. A 2D Integer Array is created to hold all the generated values from the Needleman-Wunsch algorithm. The constructor sets a couple private variables that will be used later, and the most important thing is that the Grid is instantiated. The last line in the constructor calls “populateGrid()” method that will fill the Grid with scores based on how well the local alignment of the AA sequence from species A is compared to the AA sequence from species B. In order to populate the grid, we run a nested for-loop that goes through the entirety of sequence A and sequence B. The Needleman-Wunsch algorithm looks at a cell and asks for the scores from the top, left, and corner. The top score is subtracted by two and compared to the left score subtracted by two. This score is for if the sequence was to consider that column an indel. The best of those two scores is compared to the score from the corner. The corner score is the case where it’s not an indel. However, for the case where we compare AA from sequence A to AA from sequence B, we will base our answers off of the BLOSUM62 Substitution Matrix. The comparison between amino acids took up the most code because it is a simple double switch case method, but it is between all possible amino acids. I hope to never have to write that code again. The greatest of the three scores is stored in the grid. Now, our grid is not exactly a Needleman-Wunsch table because we aren’t storing arrows. Arrows would help us delineate long chains where sequence A and B matched up, and we could get even better local alignment scores than the calculation we have currently for best score. The last method in the Grid class is a “getScore()” method that takes two integers for coordinates and returns the score at that cell.

**WPGMA Tree:**

The WPGMA Java program is also split into three classes: Main, Tree, and Node. The main purpose of this java program is to pair sequence via closest score. The closest score means the sequences are highly similar. In order to do this pairing, we have to wait for the Pairwise Alignment Java program to finish executing before running the WPGMA Java program. The output for Pairwise Alignment (Figure 2) will now be the input for WPGMA and is in the command line argument. The Main will read and store the first two lines of the file, the number of amino acid sequences and the number of outputted scores respectively. In order to make the tree more appealing, the scores were altered, meaning the best scores were represented as a lower score and vice versa. This is done so pairing the amino acid sequences together and creating the tree will be more understandable when tracing scores for each sequence. Once completed, the next step is creating an ArrayList, called “sequences”, consisting of Nodes. Each new Node will be given a name and a blank score which will be altered during pairing. The last step is having the scores and number of sequences be sent to create a 2D array called “grid”. This will be used as a template to view the aligned sequences and their scores.

Once all of it is created, pairing process will begin. The function “pair(float[][] grid)” will start by finding the minimum value in “grid”. Once found, it will store the location of the minimum value in “x” and “y” which will be used to call on the sequences. After, “sequences.get(x/y)” will call upon the the correct Nodes and set their scores by half the minimum. It is then given level, which is mainly used for printing purposes, depending on how far away the parent Node’s level is. Once the current nodes have been updated, they will then be used to create a brand new Node which is called by the name of the two nodes combined. The score will be set to 0 as well. This will be placed at the very end of the ArrayList “sequences”. The two current nodes will be sent to a new ArrayList<Node> called “pairedSequences” and removed from the previous ArrayList “sequences”.

Depending on the current size of “grid”, “grid” will now be reduced down. A for loop will be used to run through the grid. It will average the scores of the sequences based on their relation to the current Nodes. That average will be stored in an Array called “newScores”. Once all the scores have been averaged by the current Nodes, all the remaining scores will be entered into another Array called “oldScores”. These are the scores that have no relation to the current nodes. The “newScores” will be added to the end of the “oldScores” and a new grid will be created via the updated “oldScores” and the currentSize reduced by one. Now the program will call “pair(oldScores, currentSize-1)” until the currentSize is equal to three.

The reasoning why the currentSize must be less less than or equal three is because the process of finding the minimum score and then averaging the remaining two scores doesn’t necessarily require a recursive call back. However, it will conduct the same procedure: setting the scores for the current Nodes, creating a new Node, and setting its score. Once all the sequences have been paired, the updated Nodes will be stored in “pairedSequnces” and will now be ready to be printed.

Printing the WPGMA tree in another file called “tree.txt” that is stored in the command line argument requires the Nodes to be on the specific levels. This is why the Nodes have a value called level to in order to be ultrametric. Every leaf Node will be on the same level. Each node will be connected by “-” and will end with “|” to show where the parent Node is. The WPGMA guide tree created by our program is shown in Figure 3.

**Results:**

**Figures:**

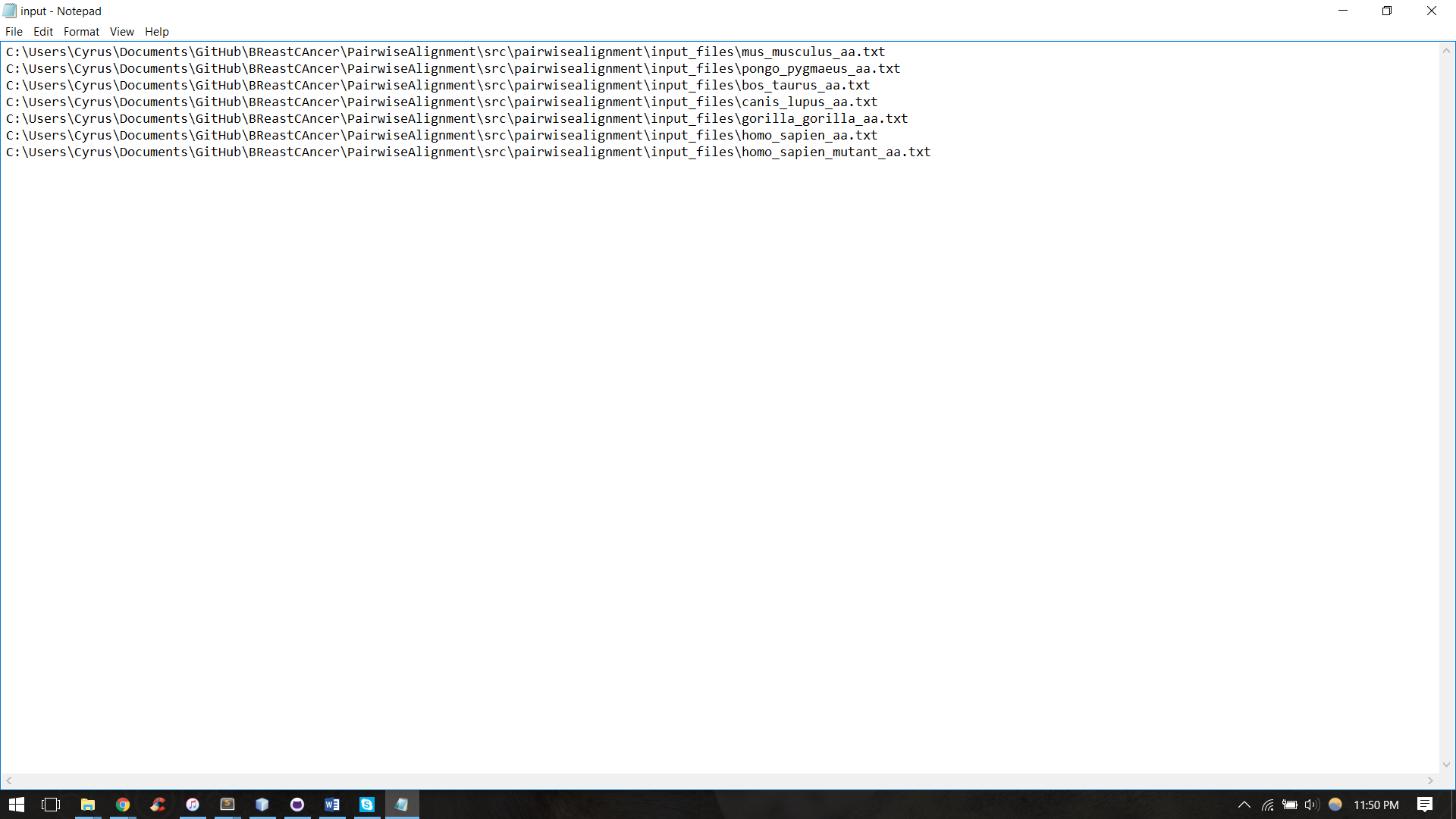


Figure 1: “input.txt” input file for Pairwise Alignment

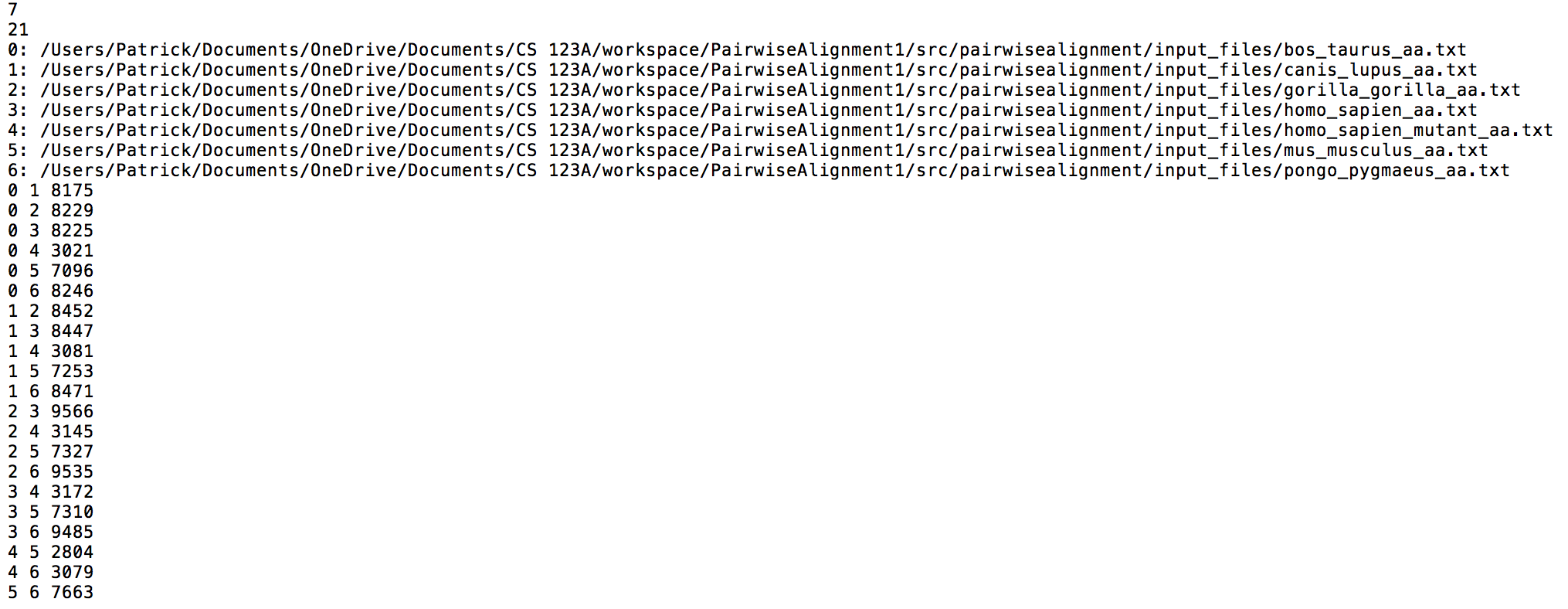


Figure 2: “output.txt” output file for Pairwise Alignment

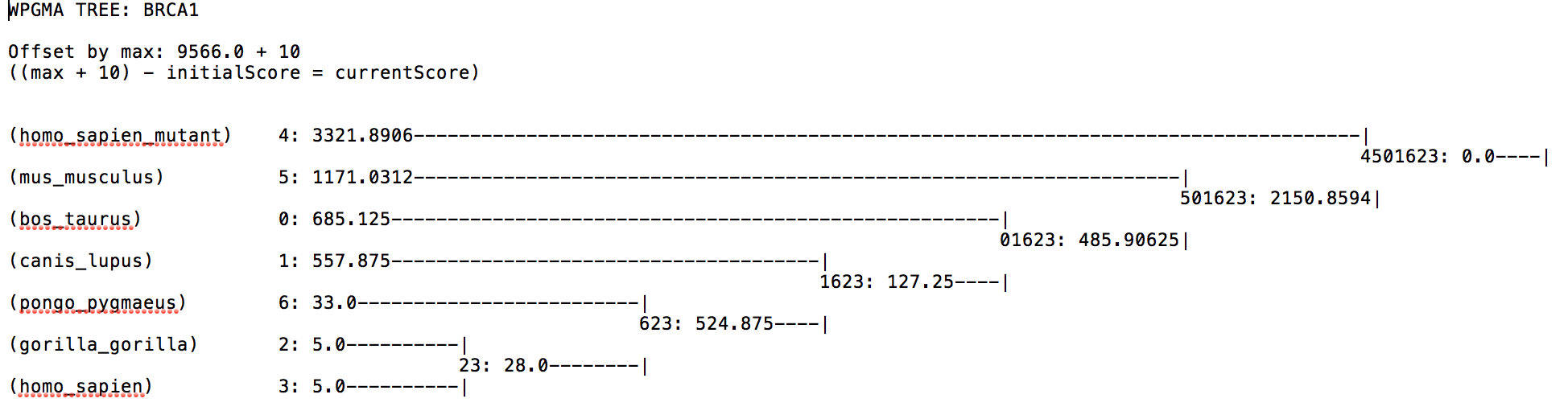


Figure 3: “tree.txt” output file for WPGMA tree

**Discussion:**

**Executive Summary:**

The 7 amino acid sequences we got were processed first in the Pairwise Alignment program. The program matched amino acid sequences and gave them scores based on the BLOSUM62 Substitution Matrix. So, from sequence Strings, we turn the data into pairs with scores attached to them highlighting the degrees of similarity between them. From there, the table is passed to the WPGMA tree builder and turned into a Grid with scores called a difference table. Then, the sequences are clustered by closets pairs and turned into a tree. The tree is printed to a text file and can be read by humans.

In our results, the WPGMA tree we built is similar to the phylogenetic tree that is produced by EMBL-EBI’s Clustal Omega. Gorillas and Modern Humans are most similar. Then, the next cluster is Orangutans. Then comes Wolves. After that is Cattle. The next one up is the Human Mutant BRCA1 gene. And trailing the pack is the Mouse sequence. The WPGMA guide tree we created does not portray the same results as the professional Multiple Sequence Alignment and Phylogenetic tree builder; however, it still shows in a sense, which BRCA1 genes are most similar to Modern Humans.

The conclusions we can draw from our project are that the free professional software that is available online is much easier to use. That being said, anyone that knows the algorithms used behind the scenes by the professional websites can also create their own version of it. The BRCA1 gene comparisons show that Modern Humans are more similar to other primates, and the mutant variation of the BRCA1 gene is more similar to House Mice.

**Data Interpretation:**

So the data shows that Modern Humans are more similar to other primates, but that much is obvious. We need to look at in what way other mammals are similar to Modern Humans. Szabo and others talk about their project comparing humans, dogs, and mice. The team found that the amino terminalis conserved across all three species. This is also true for the C-terminus. The team also found motifs in all three that they believe holds biologically important amino acids. The parts of the different genes that stay the same were mostly the parts of the gene that protect from breast cancer. The parts of the amino acids that were different from each other are most likely parts of the gene that do not help to prevent breast cancer. Even though mouse seems like the farthest away from us on our generated guide tree, it is actually still very similar in structure.

Herschkowitz and group do something else. Our data shows that House Mouse and Modern Human Mutant Gene are similar to each other. The group of scientists took 13 different mice dna samples and compared them to known breast cancer dna. From this, scientists are able to find out what is similar and different from the breast cancer and other living things. The mice that had cancer similar to human cancers showed in their bodies. This is meaningful for our project because it could mean that the mouse we compared to had similar links to the mutant amino acid chain.

**Improvements:**

After creating and executing the Pairwise Alignment and WPGMA programs, we decided to compare our generated tree with an online phylogenetic tree generator. We used a website called www.phylogeny.fr which can generate a phylogenetic tree when inputed FASTA formated files. The tree generated showed to be similar to ours in terms of neighbor pairing, however it still has its major differences. In figure 4, mice is the outgroup in the pairings. However in our WPGMA tree, it shows that the mutated human sequence is a part of the outgroup. This may be due to the scoring for humans and mutated human sequence being low, causing the mutated human sequence to be casted as the outgroup. Although they are from the same species, the mutated human sequence is significantly shorter than the human sequence. To improve on this, we would need to ensure those species that share very similar sequences will be paired closely together.

Another improvement we can work on is the execution of pairing. The clustering the neighbors together brought one species closer to another when they are actually closer to a different species. Figure 3 shows us having wolves more closely related to humans than cattle whereas the generated phylogenetic tree has both cattle and wolves the same distance away from humans. This can be due to an error when regrouping clustered sequences back into the grid and we can use better techniques in calculating and resizing the grid after pairing

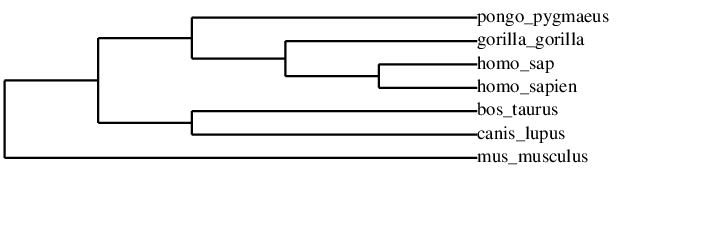


Figure 4: Phylogenetic Tree generated by www.phylogeny.fr

**Next Steps:**

Now that we have an understanding of how closely related the BRCA1 gene is to other species in respect to the Modern Human, the next step is to determine the outcome of the gene will or will not be expressed in other species. Since Humans are susceptible to breast cancer, we will want to see if those species closely related to Humans can be inflicted by breast cancer as well. If the BRCA1 gene in any other species were to be mutated, we would want to know the likelihood of the species to develop breast cancer. We want to know if any alterations in a certain gene that can cause an effect in humans can do the same in other species.

**References:**

Easton, D. F., Deffenbaugh, A. M., Pruss, D., Frye, C., Wenstrup, R. J., Allen-Brady, K., ... & Goldgar, D. E. (2007). A systematic genetic assessment of 1,433 sequence variants of unknown clinical significance in the BRCA1 and BRCA2 breast cancer–predisposition genes. *The American Journal of Human Genetics*, *81*(5), 873-883.

Herschkowitz, J. I., Simin, K., Weigman, V. J., Mikaelian, I., Usary, J., Hu, Z., ... & Backlund, M. G. (2007). Identification of conserved gene expression features between murine mammary carcinoma models and human breast tumors. *Genome biology*, *8*(5), 1.

Szabo, C. I., Wagner, L. A., Francisco, L. V., Roach, J. C., Argonza, R., King, M. C., & Ostrander, E. A. (1996). Human, canine and murine BRCA1 genes: sequence comparison among species. *Human molecular genetics*, *5*(9), 1289-1298.

Welcsh, P. L., Owens, K. N., & King, M. C. (2000). Insights into the functions of BRCA1 and BRCA2. *Trends in Genetics*, *16*(2), 69-74.