ENS 210 - Computational Biology
Fall 2023-2024

Evolutionary history of TMEM216 protein, with respect to its homologous sequences, phylogenetic tree and mutations

Authors:
Samet Mert -23511
Dila Karataş -28852
Nil Özde Özden -30802
Türkan Doğa Gizer - 29072
Mehmet Barış Tekdemir -29068

A. INTRODUCTION

Proteins are molecules that are large and complex and are vital to many body functions (*What are proteins and what do they do?* 2021). They are essential for the function, structure and regulation of the body's tissues and organs and perform the majority of the work in the cells (*What are proteins and what do they do?* 2021). The mutations in the proteins may change their function, which may cause the protein not be able to bind to certain regions or bind to some regions very well (Agnihotry et al., 2022). The main target of the project is the Transmembrane protein 216 whose mutations may cause Joubert Syndrome 2 also known as Cerebello-oculorenal Syndrome 2 (TMEM 216 protein, n.d.). Joubert Syndrome is a genetically heterogeneous autosomal recessive disorder characterized by MTS, hypotonia, retinal dystrophy, respiratory abnormalities, renal diseases and endocrine abnormalities (Joubert Syndrome 2, n.d.). The inheritance pattern for Joubert Syndrome is autosomal recessive, meaning that each cell's two copies of a gene have a mutation (*Joubert syndrome* 2017). Although the parents of a person with an autosomal recessive condition each have one copy of the mutated gene, they typically don't show the signs and symptoms of the disease (*Joubert syndrome,* 2017).

The gene coding transmembrane protein 216 is called TMEM216 (TMEM 216 protein, n.d.). TMEM 216 is a transmembrane protein that spans the cell membrane, enabling communication between the interior and exterior of the cell (OMIM, n.d.). TMEM 216 protein consists of 87 amino acids and 2 transmembrane domains (OMIM, n.d.). With respect to its location in the cells, TMEM216 protein plays a role in protein trafficking, lipid metabolism, and cellular signaling pathways (OMIM, n.d.). The mutant protein can alter the functions of the protein and lead to various disorders in the body (OMIM, n.d.). The aim of the project is to reveal specific aspects of the TMEM 216 protein through a computational approach; This was achieved by investigating the functions of the TMEM 216 protein, identifying conserved regions, and examining the effects of these regions on Joubert Syndrome 2.

B. MATERIALS AND METHODS

The second part of the project has started with choosing the TMEM 216 gene and Joubert Syndrome caused by the mutations of this gene. Joubert Syndrome is a rare hereditary disease which is inherited in a Mendelian manner. For this project, TMEM216 gene and Joubert Syndrome 2 was chosen among a wide variety of diseases, due to its autosomal recessive inheritance pattern. The mutations that cause Joubert Syndrome 2 occur on TMEM216, a transmembrane protein that is vital for cell communication and signaling. In Joubert Syndrome 2, brain development is impaired and neurological symptoms are observed. The UniProt database was used to obtain the TMEM216 protein's sequence for Homo Sapiens. Sequences of homologues of the protein were found using BLASTp against 100, 250, 500, 1000 and 5000 hit number sequences.

It was decided to exclude files with 100 and 250 sequences due to small coverage of the data. It was also decided to exclude the 5000 sequences file due to the presence of a paralogous gene in Homo Sapiens. Files with 1000 sequences were excluded too due to the complexity of the analysis related to time consumption of the tree forming of this data. In the end, it is decided to make an analysis on 500 hit data. Homologous sequences were aligned with the MEGAX tool and MUSCLE algorithm using a 500-sequences fasta file. As a result, the analysis was carried out using a sequence file of 500 sequences containing 4 different homo sapiens isoform sequences for the TMEM216 protein. Parameters for MUSCLE alignment used as the following; for Gap Penalties parameters: Gap Opening: -2.90, Gap Extend: 0.00, Hydrophobicity Multiplier: 1.20, for Memory/Iterations parameters: Max Memory in MB: 2048, Max Iterations: 16, and for Advanced Options: Cluster Method(Iterations 1,2): UPGMA, Cluster Method(Other Iterations): UPGMA, Min Diag Length (Lambda): 24. These are used as default from the program.

The resulting alignment for 500 sequences was then saved as a meg and a fasta file. Afterwards, the phylogenetic tree of the aligned sequences was drawn via MEGAX. The phylogenetic tree was constructed using the Neighbor-Joining method and the parameters for phylogenetic tree construction on MEGAX was specified in **Figure 1**. The created bootstrap phylogenetic tree was saved in MEGAX as Newick and pdf files. The Newick file downloaded from MEGAX was then used to draw the phylogenetic tree of 500 homologous sequences in FigTree. Since the phylogenetic tree constructed in FigTree is in an unrooted format, Midpoint Rooting was applied onto the phylogenetic tree.

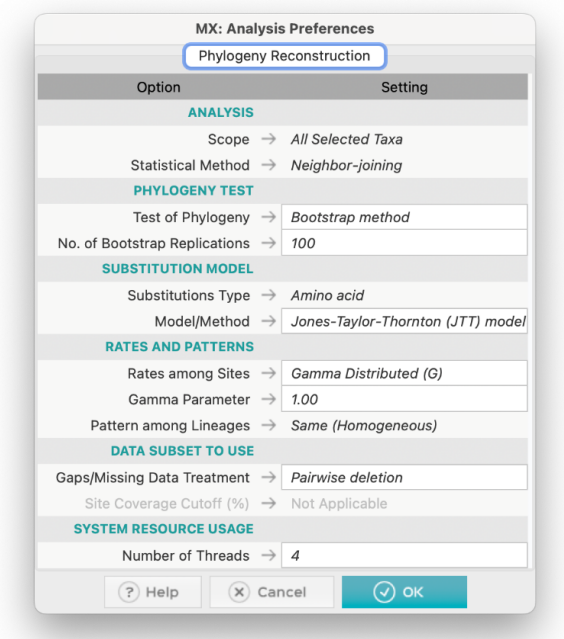

Fig.1 The parameters used to create a phylogenetic tree of 500 sequences on MEGAX

The aligned sequences then were used to calculate conservation scores for each amino acid position in the sequence of 500-sequences file. Conservation scores were calculated with a python code by using the "500_aligned.fas" file **(Fig.2)**. For each position, the amino acid with the highest number of occurrences among all aligned sequences was used to obtain the consensus sequence. The frequency of amino acids in the consensus sequence was calculated and this was treated as conservation scores for the respective position. The obtained consensus sequence and the position and conservation scores of each amino acid were pushed to Github in tsv file format to be used in drawing the conservation plots.

```
filename = "../conservation_score/500subtree_aligned_fasta.fas"
sequence_dict = fastareader(filename)
alignment_sequences = list(sequence_dict.values())
amino_acids = ["A", "R", "N", "D", "C", "Q", "E", "G", "H", "I", "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V"]

consensus_sequence = ""
consensus_scores = {}
for position in range(len(alignment_sequences[0])):
    amino_acid_percentages = dict.fromkeys(amino_acids, 0)
    amino_acids_at_position = ""
    for seq in alignment_sequences:
        amino_acids_at_position += seq[position]
    for amino_acid in amino_acids:
        count_amino_acid = amino_acids_at_position.count(amino_acid)
        amino_acid_percentages[amino_acid] = count_amino_acid / len(alignment_sequences)
    amino_acid_percentages = dict(sorted(amino_acid_percentages.items(), key=lambda x: x[1], reverse=True))
    consensus_amino_acid = list(amino_acid_percentages.keys())[0]
    max_amino_acid_score = list(amino_acid_percentages.values())[0]
    consensus_sequence += consensus_amino_acid
    consensus_scores[position] = {consensus_amino_acid: max_amino_acid_score}

print("Consensus Sequence:", consensus_sequence)
print("\nAmino Acid Scores:")
for pos, data in consensus_scores.items():
    print(f"Position {pos + 1}: {data}")

conservation_scores = [sum(data.values()) for data in consensus_scores.values()]
import matplotlib.pyplot as plt

plt.hist(conservation_scores, bins=20, color='blue', edgecolor='black')
plt.xlabel('Conservation Score')
plt.ylabel('Number of positions')
plt.title('Conservation Score Histogram')
```

Fig.2 Conservation score and consensus sequence calculation code

The phylogenetic tree was then pruned to create the subset tree containing branches where Homo Sapiens sequences were found **(Fig.3)**. The phylogenetic tree consisted of two main clades. Since all Homo Sapiens sequences were found in the same clade, the clade that is without Homo Sapiens sequences was removed from the phylogenetic tree. From the subtree obtained after this process, protein names with species information are collected in a file which is in Newick format. A Python code used for obtaining the sequences of these proteins in a fasta file with the searching of the headers in 500-sequences fasta file, and header information taken from the previously mentioned Newick file. **(Fig.4)**. The resulting fasta file was aligned again with the MUSCLE algorithm via MEGAX to create a tree, and the phylogenetic tree was drawn via MEGAX. The drawn tree was rerooted with Midpoint rooting in FigTree, and the phylogenetic tree created for further analysis was saved in Pdf and Newick format and pushed to Github.

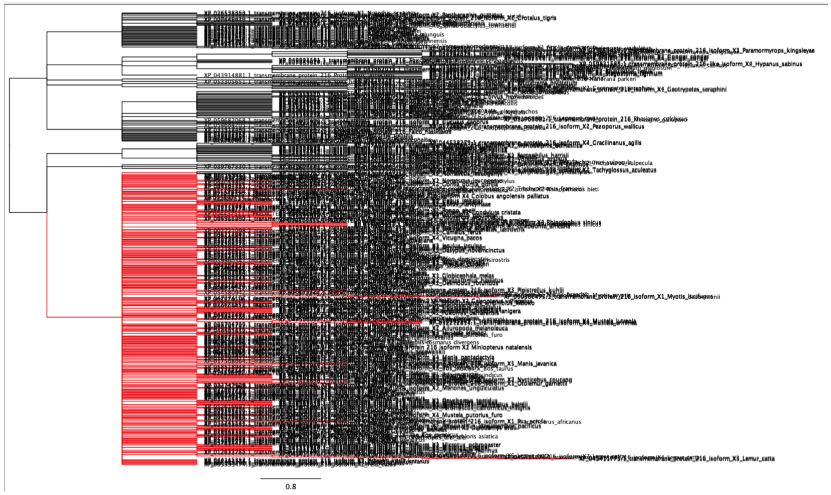

Fig.3 Red branches indicates the subtree of 500 sequences tree

```python
import re

def extract_xp_words_from_file(filename):
    try:
        with open(filename, 'r') as file:
            newick_string = file.read()
            pattern = r'[XN]P_\d+\.\d+'
            xp_matches = re.findall(pattern, newick_string)
            return xp_matches
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return []
filename = "input.nwk"
xp_words = extract_xp_words_from_file(filename)
def extract_sequences_from_fasta(input_fasta, xp_identifiers, output_fasta):
    try:
        with open(input_fasta, 'r') as input_file, open(output_fasta, 'w') as output_file:
            in_xp_list = False
            for line in input_file:
                if line.startswith('>'):
                    identifier = line.strip()[1:]
                    index=identifier.find(" ")
                    aa=identifier[0:index]
                    in_xp_list = identifier[0:index] in xp_identifiers
                    if in_xp_list:
                        output_file.write(line)
                elif in_xp_list:
                    output_file.write(line)
    except FileNotFoundError:
        print(f"Error: File '{input_fasta}' not found.")
input_fasta = "input.fasta"
output_fasta = "output.fasta"
extract_sequences_from_fasta("500.fas", xp_words, "500_pruned.fas")
```

Fig.4 The python script to retrieve the subtree that is stated in Figure 3.

In order to be able to better focus on the relevant sequence, the phylogenetic tree was pruned a second time to create a more specific subset tree. The phylogenetic tree consisted of two main branches. Since all Homo Sapiens sequences were found on the same branch, the branch containing no Homo Sapiens sequences was removed from the phylogenetic tree. Also, Bootstrap values were taken into consideration when pruning the pruned tree. Species with a Bootstrap value less than 0.7 were removed from the analysis, and the fasta file of the considered species used for realignment and drawing of more specific phylogenetic tree in MEGAX and then again the tree formation with Midpoint rerooting in Figtree. Python code used in this step, and the code pushed to Github **(Fig. 4)**. Also, the generated phylogenetic tree was saved in Pdf and Newick format for further analysis and pushed to Github. The final phylogenetic tree that is used in the rest of the project is shown in **Figure 5**.

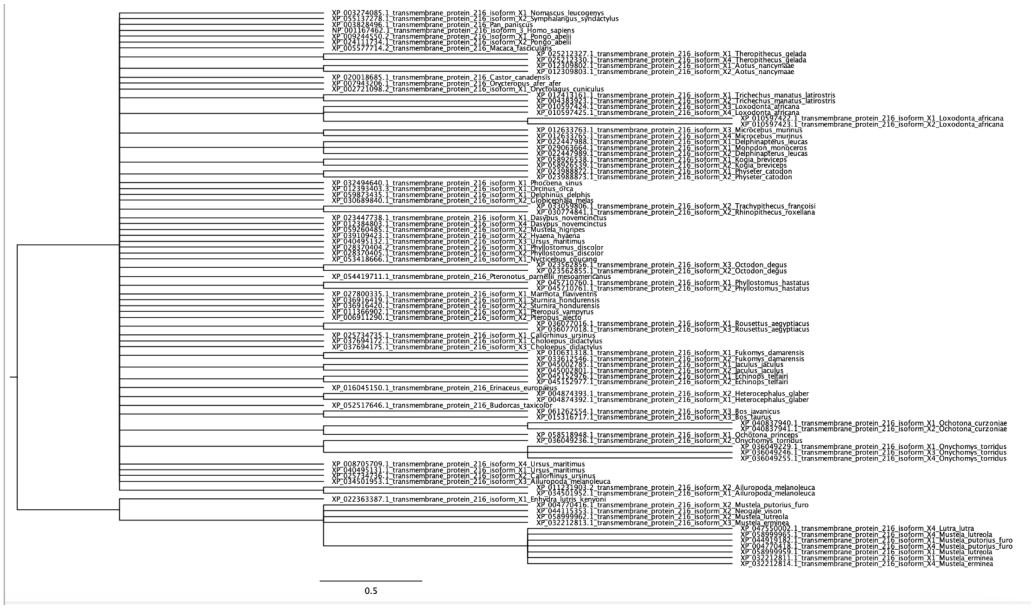

Fig.5 Final version of the phylogenetic tree after 2 subtree construction

Subsequently, single point mutations were obtained from two different sources for cross-checking of single point mutations. The first source used was scientific articles on Joubert Syndrome 2, which identified mutations with confirmed pathogenic effects on the TMEM216 protein. The ClinVar database which was in the gnomAD database was used as the second source, reporting relationships between variations and pathogenicity. Also gnomAD, a database where population frequencies per positions and variants in these positions are reported used in the analysis for making classification upon it. The used resources and the data obtained were pushed to Github.

Mutations that were definitely pathogenic were taken from clinical articles and pushed to Github as a txt file named paper_mutations.txt. After the subtree creation process, from the last tree, Homo Sapiens isoform 3 was determined as the original sequence. The protein sequence of

Homo Sapiens isoform 3 has been deposited on Github in fasta format. Afterwards, for mapping the original sequence and aligned sequences of the three different trees, a Python code named map_mutation_positions.py is used **(Fig.6).** The function in this code is imported in a different Python code which is named known_mutations.py **(Fig.7).** This second code is used in the aim of calculating the threshold scores of conservation with respect to the three different trees and three different scores saved in a file named conservation_thresholds.txt. Threshold scores of conservation determined according to the known pathogenic mutations. These known Pathogenic mutations are obtained from different sources as mentioned in the previous paragraph, and used as an input file in the code.

```python
import pandas as pd

def fastareader(filename):
    seqDict = {}
    infile = open(filename, 'r')
    for line in infile:
        if line[0] == ">":
            header = line.strip()[1:]
            seqDict[header] = ""
        else:
            seqDict[header] += line.strip()
    return seqDict

def map_mutation_position(aligned_filename):
    filename = "tmem216_proteinSequenceIso3.fas"
    original_sequence = list(fastareader(filename).values())[0]
    sequence_dict = fastareader(aligned_filename)
    aligned_sequence = sequence_dict['NP 001167462.1 transmembrane protein 216
    originalPos_alignedPos_dict = {}
    modified_aligned = aligned_sequence
    for i in range(len(original_sequence)):
        aa = original_sequence[i]
```

Fig.6 The python code for mapping the original sequence with the sequences in aligned files. Saved to Github as “map_mutation_positions.py”

```python

def main():
    filename = "paper_mutations.txt"
    mutations = open(filename, 'r').read()
    mutations = mutations.split(",")
    position_list = []
    #
    filename = "tmem216_proteinSequenceIso3.fas"
    original_sequence = list(fastareader(filename).values())[0]
    #
    for mutation in mutations:
        substr = mutation[3:]
        if substr[1].isalpha(): #Then the mutation position is a 1-digit number
            position = substr[0]
        elif substr[2].isalpha(): #Then the mutation position is a 2-digit number
            position = substr[:2]
        else: #Otherwise
            position = substr[:3]
        if int(position) <= len(original_sequence):
            position_list.append(int(position))
    aligned_filename = "500_aligned.fas"
    position_dict = map_mutation_positions.map_mutation_position(aligned_filename)
    conservation_scores = pd.read_csv("output.tsv", sep = "\t")
    cons_score_list = mutation_pos_cons_score(position_list, position_dict, conservation_scores)
    #
```

Fig.7 The python code for calculating the threshold scores of conservation with respect to the three different trees. This code was saved to github as “known_mutations.py”.

For the mutations whose clinical significance is unknown, the classification is applied with the information of three different aligned sequences data. It is performed with a Python code which is pushed to Github as gnomadVarAnalysis.py **(Fig.8)**. For the classification purposes, three different threshold scores of conservation are used to determine if the variant is Pathogenic or not, then label them as True and False according to the question of "Is variant Pathogenic?". The unknown variants to classify are retrieved from gnomAD as previously mentioned, and for this purpose the file named gnomAD_all.csv is used. The classification results are saved in the csv format, and named as classification_output.csv, classification_output2.csv, and classification_output3.csv related to three different aligned sequences data. These classification results are reported in the results section.

```
variants = pd.read_csv("gnomAD_all.csv")
cols_to_keep = ['HGVS Consequence', 'VEP Annotation', 'ClinVar Clinical Significance', 'Allele Count', 'Allele Number']
variants = variants[cols_to_keep]

missense_variants = variants.loc[variants['VEP Annotation'] == 'missense_variant']
missense_variants = missense_variants.loc[missense_variants['ClinVar Clinical Significance'].isnull()]

position_list = []
for i in list(missense_variants.index):
    row = missense_variants.loc[i]
    mutation = row[0]
    substr = mutation[5:]
    if substr[1].isalpha(): #Then the mutation position is a 1-digit number
        position = substr[0]
    elif substr[2].isalpha(): #Then the mutation position is a 2-digit number
        position = substr[:2]
    else: #Otherwise
        position = substr[:3]
    position_list.append(int(position))

conservation_thresholds = open("conservation_thresholds.txt", 'r').read().strip().split(" ")
conservation_thresholds = [float(score) for score in conservation_thresholds]

aligned_filename = "500_aligned.fas"
position_dict = map_mutation_positions.map_mutation_position(aligned_filename)
conservation_scores = pd.read_csv("output.tsv", sep = "\t")
cons_score_list = known_mutations.mutation_pos_cons_score(position_list, position_dict, conservation_scores)
threshold1 = conservation_thresholds[0]
isPathogenic = []
for score in cons_score_list:
    if score >= threshold1:
        isPathogenic.append(True)
```

Fig.8 The python code for determining the pathogenicity of mutations

Finally, a t-test, which is a statistical test used to compare the means of two groups and determine if they are significantly different from each other was performed to determine whether there was a significant difference in allele frequencies between variants classified as pathogenic and nonpathogenic. t-test is performed with a Python code named as ttest.py. **(Fig.9)**. Results of the t-test, determined p-values are printed and then pushed to Github as results_ttest.txt. The findings are reported in the results section.

```python
from scipy import stats as st
import pandas as pd


outputList=["classification_output.csv","classification_output2.csv","classification_output3.csv"]
for i in outputList:
    df = pd.read_csv(i)

    a = df.loc[df['isPathogenic'] == True, 'Allele_frequency'].to_numpy()
    b = df.loc[df['isPathogenic'] == False, 'Allele_frequency'].to_numpy()

    pvalue = st.ttest_ind(a=a, b=b, equal_var = True).pvalue

    print(f"P-value for {i}: {pvalue}")
```

Fig.9 The python code for statistical t-test

C. RESULTS

In this project, human TMEM216 protein, a transmembrane protein, was examined in terms of its homologous sequences, conserved regions and possible mutations. The protein sequence file taken from UniProt was used to identify the homologous sequences using BlastP, as is further mentioned in the Materials and Methods part. 500 homologous sequences were aligned using the MUSCLE algorithm via MEGAX tool. **Figure 10** shows a small portion of the 500 sequence alignment. The entire alignment file has been uploaded to Github.

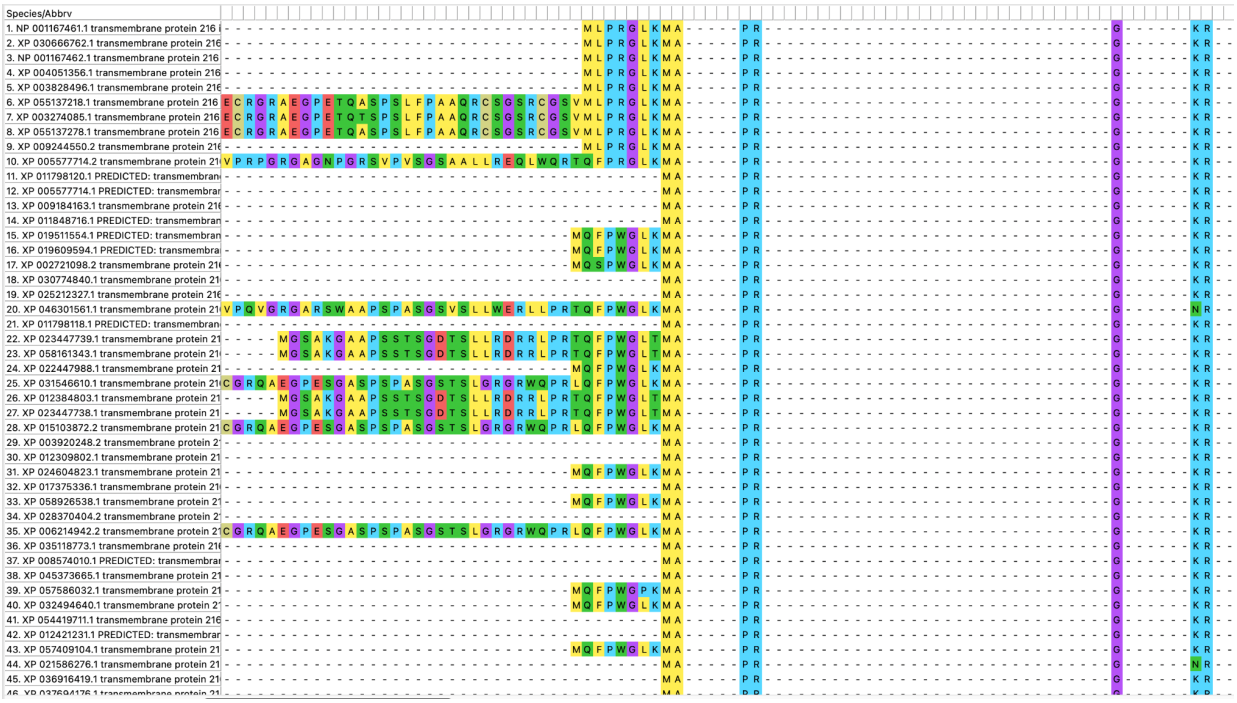

Fig.10 The alignment of 500 homologous sequences

The aligned protein sequences file were used to construct a phylogenetic tree by the Neighbor-joining method via MEGAX, and Midpoint rerooting applied via Figtree. It is observed that all four isoforms of Homo Sapiens TMEM216 proteins are cumulated in the same clade in the tree, and highlighted as red in the **Figure 11.** Also, it can be seen in the same figure that the three first branched to two, and in the branch below, the second branch which contains the Homo Sapiens isoform taken into consideration to create a new tree. The second obtained tree can be seen in **Figure 12**. After that, a second subtree is taken into consideration to create a new tree from the tree in **Figure 12**, and this tree can be seen in **Figure 13.** Different from the first and second tree, it is determined that in the third tree which showed in **Figure 13,** isoform 1, 2, and 4 of TMEM216 protein in Homo Sapiens were not observed after the applied methods mentioned in the Materials and Methods section to construct the third tree. Therefore, it can be concluded that after two subset tree construction, there is only one isoform of human TMEM216 protein, isoform 3, in the final phylogenetic tree.

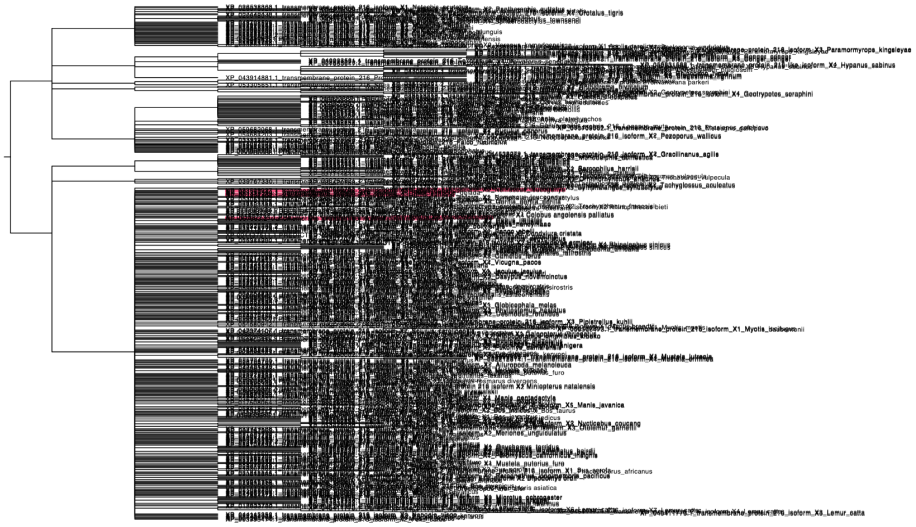

Fig. 11 The phylogenetic tree of 500 homologous sequences. The parts that highlighted in pink are the Homo Sapiens isoforms of human TMEM216 protein

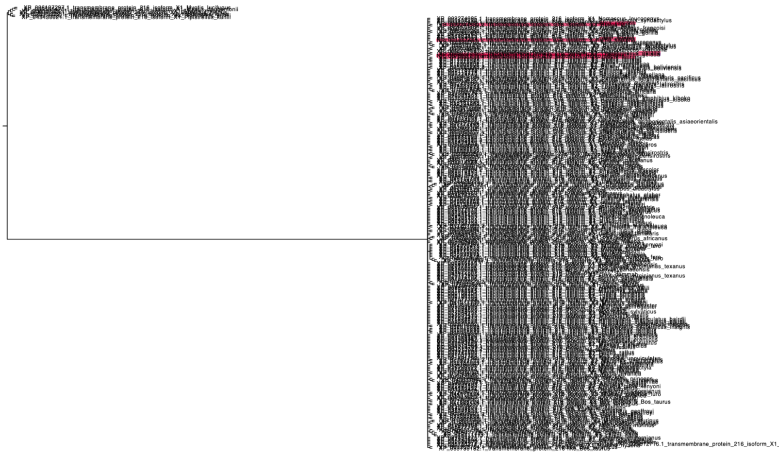

Fig. 12 The phylogenetic tree of the first subset tree of the 500 homologous sequences. The parts that highlighted in pink are the homo sapiens isoforms of human TMEM216 protein

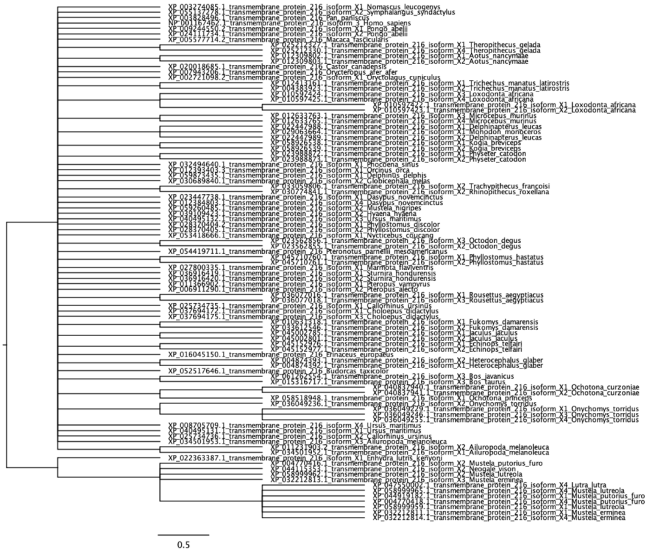

Fig.13 The phylogenetic tree of the second subset tree of the 500 homologous sequences.

Conservation scores for each amino acid position in the sequences and consensus sequences for all of the alignment files were calculated one by one. The overall value distribution of conservation scores using three different alignment outputs was shown in the **Figure 14, 15** and **16** below. Also, the consensus sequences of each alignment file was shown in **Figure 17, 18** and **19** below and conservation scores per position scatter plot with a line plot overlaid for each alignment file was shown in **Figure 20, 21** and **22** below.

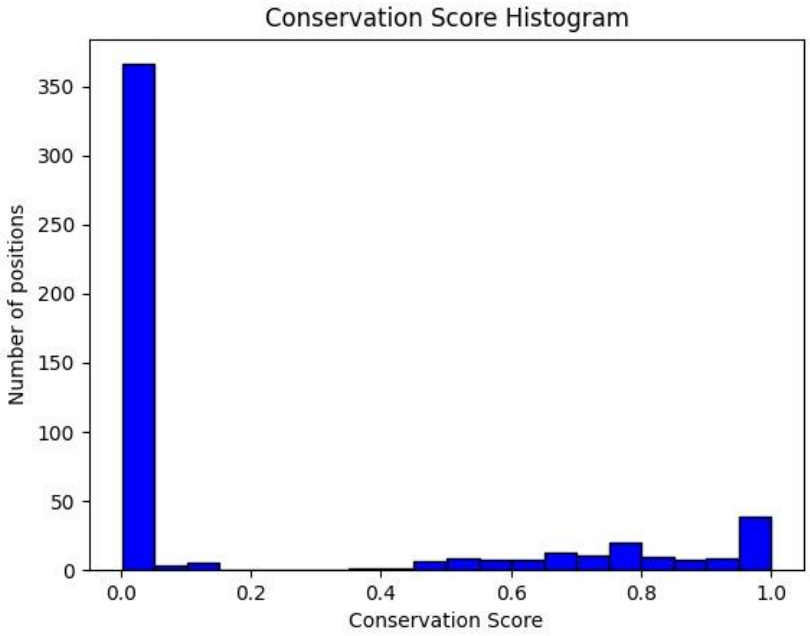

Fig.14 Distribution of the 500 sequences' alignment conservation scores

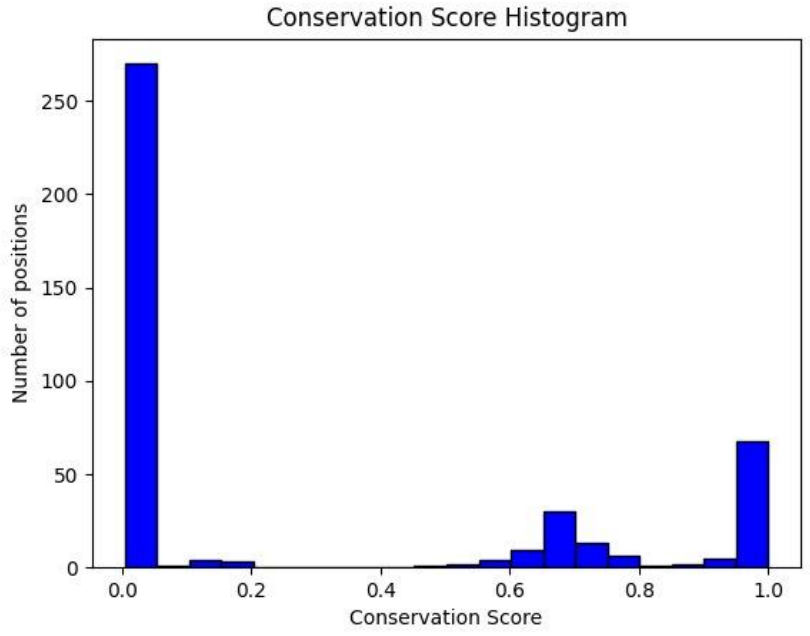

Fig.15 Distribution of conservation scores for the first subtree constructed from 500-sequences tree

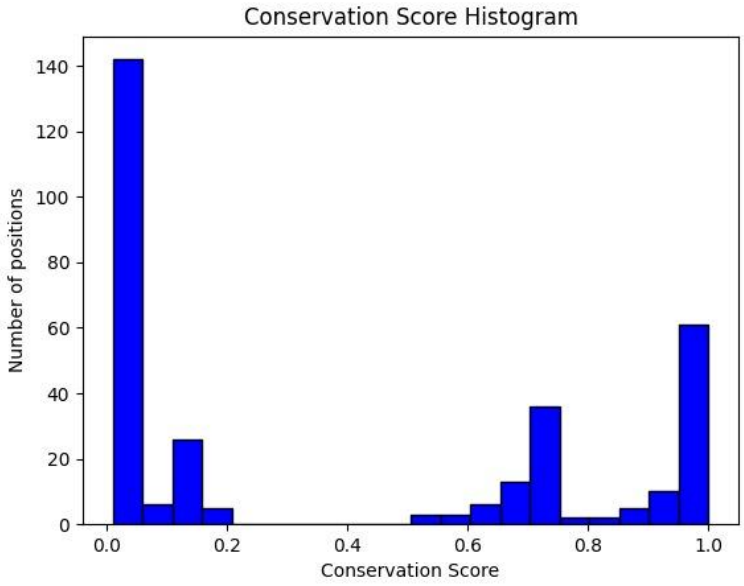

Fig.16 Distribution of conservation scores for the second subtree

```
Consensus Sequence:
MRGGEAARCHMLQTSNYSLVLSLQFLLLFYDLFVNSFSELLRLAPVIQLVLFIIQDIAILFNVIIIFLMFFNTGAFQAGL
RPGLFTRGRGGRGGARTAAPPSTSGSASRPRGRWLPRMQFPWGLKMAALFVFPRASATGKGRPARAKEGPPAAGFRGNAN
AAALLGEEPEVCKRKIAILGVLSSTPLEILFFLNGWYYATYFLLELFIFLYKACLFTARQKSLQALTFVLAGLLLPYPTA
NLVLDVVMLLLYLGIEVIRLFFGQADQAEMRRLQKLCDTGTKGNLCQRKMPLGISVALTFPSAMMASYYLLLQTYVLRLE
AIMNGILLFFCGSELLLEVGQDLTLAAFSRQCRAEGAKAIRRGRCRIRGICRTSRACVLSAGFLWTRRRSSSPSSPTVAA
IMLLKGSDRQKSKTPAVANQLPVPSGGMCMKKDEQRRLLLYCYFWTDCQANRRRGSPILVRSQVGPVVYFKICLGQPNES
RTAAERSSRSQTCAEKMRLGSKRCAPAHQKLLIGL
```

Fig.17 Consensus sequence of the 500 sequences' alignment

```
Consensus Sequence:
MVQCESHYGERELFSATKMGKLGSCLRPRGPALRERPLFASRSAAETPVKAVCVHVRHPAHARGAPRRARRQGGITRGRQ
RRGSAEGAAPSPASGSMSLLRERRLPRMQFPWGLKMAPRGEVLEVCKRLSSTPLEILFFLNGWYYATYFLLELFIFLYKA
LTFVLAQKSLQLMALERFGLLLPYPTANLVLDVVMLLLYLGIEVIRLFFGQGKQATWRRLQKLCLTGTKGNLCQRKMPLG
ISVALTFPSAMMASYYLLLQTYVLRLEAIMNGILLFFCGSELLLEYGQDLRCKHCSPWPVGQDLTLAAFSRYCCGGSISC
PRRCRALGPFRGTRRSWSASYQSEAGILGRSIVKRQHFPVVYFKSMLPDDCAEAPGSNQAQRNARRILTSSSGCGTHAPT
ARTYSPRFSLTGCSAFPQH
```

Fig.18 Consensus sequence of the first subtree

```
Consensus Sequence:
MGGHGRLRADIRARATCGRARARLALGMTTECRGRAGSPEGAASPSLFGAAPRRRERRPTSMQFPWGLKMAPRGEVLEVC
KRLSSTPLEILFFLNGWYYATYFLLELFIFLYKAGLLLPYPTANLVLDVVMLLLYLGIEVIRLFFGQGKQATWRRLQKLC
LTGTKGNLCQRKMPLGISVALTFPSAMMASYYLLLQTYVLRLEAIMNGILLFEVQNFCGSELLLEPGQVLTLAAFSPQHM
RLTAPALRSGGSMWSASDRIEAGILGRSIVKRQHFPVVYFKISNRRQLLRGKGMLMDCLFCLRWLTCLGKMFLGAKRCGT
```

Fig.19 Consensus sequence of the second subtree

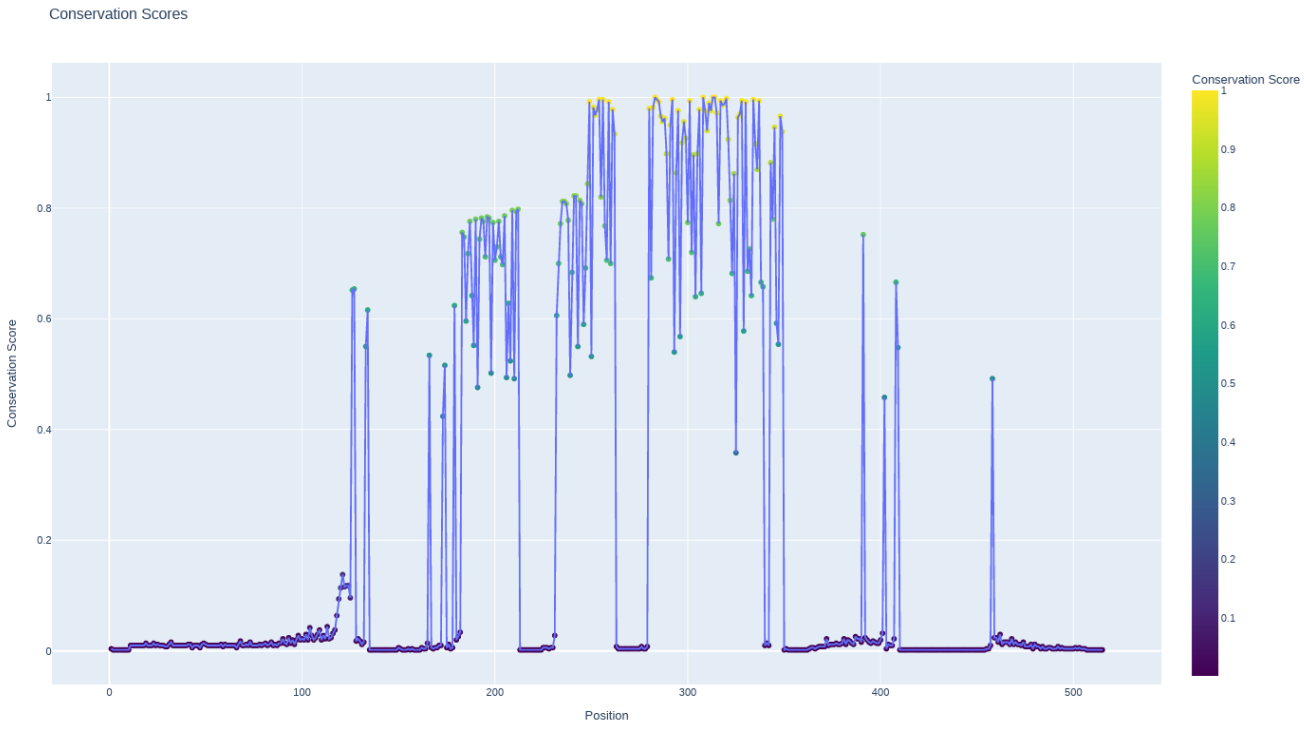

Fig.20 Conservation Scores per position scatter plot with a line plot overlaid for 500 sequences' alignment

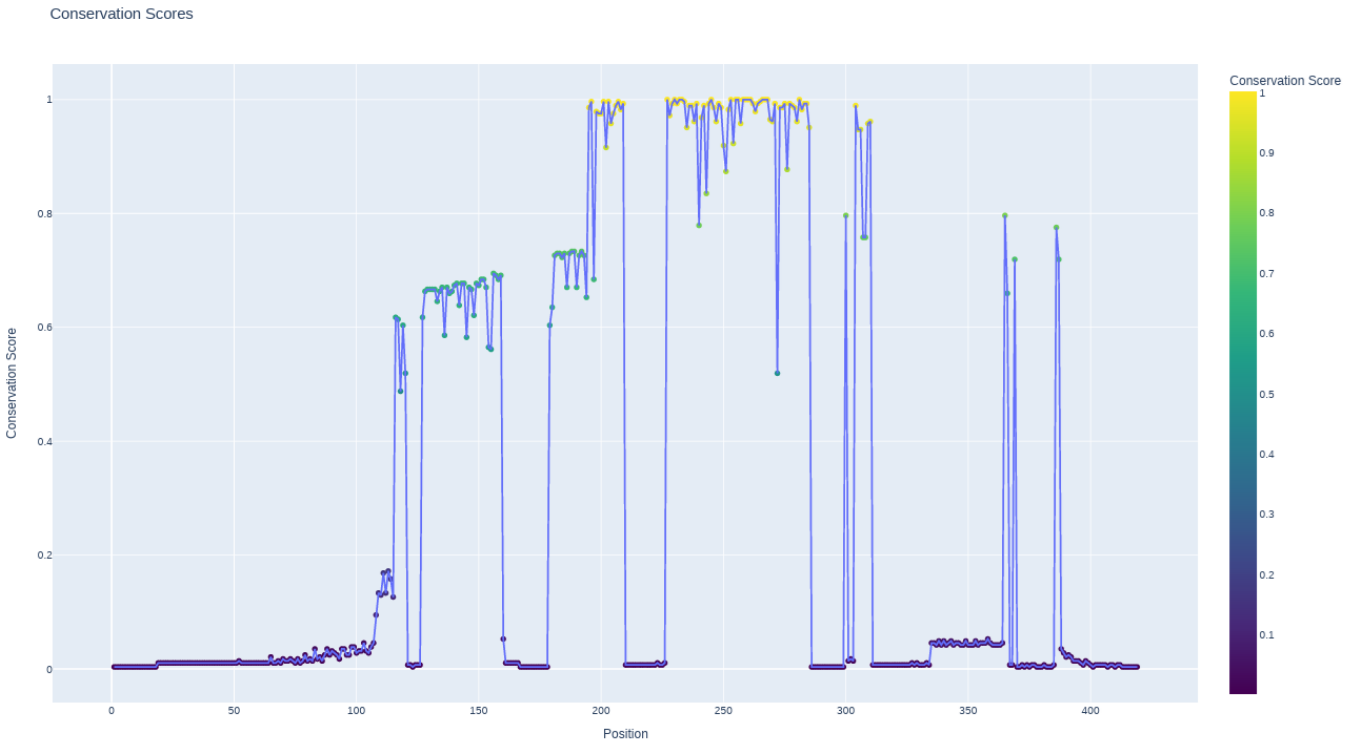

Fig.21 Conservation Scores per position scatter plot with a line plot overlaid for first subtree

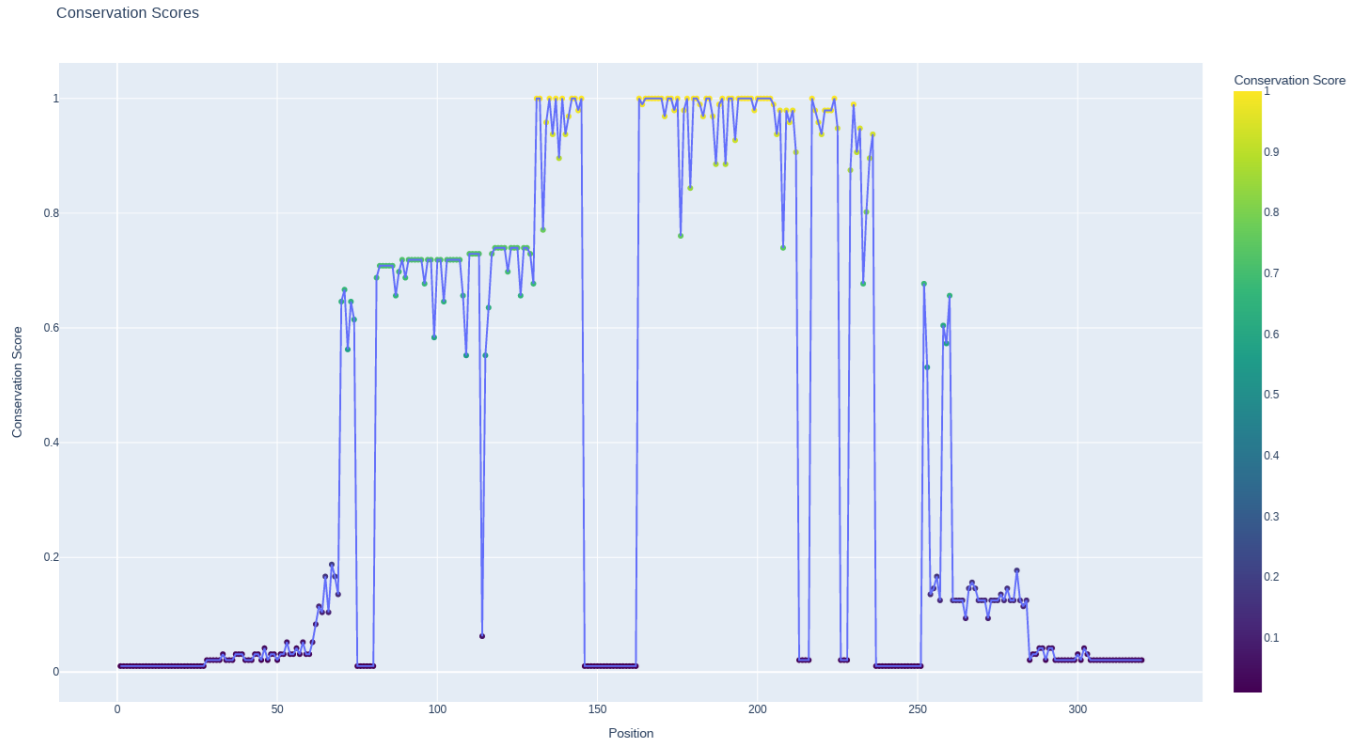

Fig.22 Conservation Scores per position scatter plot with a line plot overlaid for first subtree

The three different threshold scores of conservation which were used for the determination of unknown mutations as Pathogenic or not were shown in **Figure 23.**

```
For 500 alignment: 0.8074285714285713
For first subtree: 0.8401002506265665
For second subtree: 0.8705357142857143
```

Fig.23 Three different threshold scores of conservation

51 of the variants classified as pathogenic, and 76 as nonpathogenic in the first 500 hits data. 56 of the variants classified as pathogenic, and 71 as nonpathogenic in the subtree data. 57 of the variants classified as pathogenic, and 70 as nonpathogenic in the pruned subtree data. Small part of information of pathogenicity per position in the three different data which was determined with the help of the scores in **Figure 23** were shown in **Figure 24, 25** and **26**.

		HGVS Consequence	cons_score	isPathogenic	Allele_frequency
1	85	p.Leu2Arg	0.114	False	7.232666192203764e-07
2	90	p.Arg4Trp	0.116	False	7.226812050853631e-07
3	91	p.Arg4Gly	0.116	False	1.953226730558883e-06
4	94	p.Gly5Arg	0.118	False	2.1679527444113793e-06
5	99	p.Leu6Met	0.118	False	1.445308096326894e-06
6	100	p.Leu6Pro	0.118	False	1.8227951469902007e-06
7	103	p.Met8Thr	0.652	False	1.8163192653351836e-06
8	104	p.Ala9Ser	0.654	False	1.8158707100054476e-06
9	106	p.Ala9Glu	0.654	False	1.4464892981489276e-06
10	109	p.Arg11Pro	0.616	False	1.8162664826183299e-06
11	208	p.Gly12Val	0.534	False	7.23194436609838e-07
12	209	p.Gly12Asp	0.534	False	7.23194436609838e-07
13	215	p.Lys13Glu	0.424	False	7.230835754458172e-07
14	216	p.Lys13Arg	0.424	False	1.2006300906715843e-06
15	217	p.Arg14Gly	0.516	False	7.23053251425861e-07
16	219	p.Arg14Leu	0.516	False	2.4012198196683916e-06

Fig.24 Classification result of 500 aligned file

76	562	p.Thr78Ala	0.9719298245614036	True	4.793148263031543e-06
77	563	p.Thr78Arg	0.9719298245614036	True	6.570561257342603e-06
78	564	p.Gly80Glu	1.0	True	1.5933563413989032e-06
79	565	p.Asn81Lys	0.9929824561403509	True	1.5930060661671e-06
80	566	p.Cys83Tyr	1.0	True	3.185098833616807e-06
81	572	p.Lys86Gln	0.9894736842105264	True	3.6009996374993697e-06
82	576	p.Pro88Thr	0.9614035087719298	True	6.843043840644669e-07
83	579	p.Leu89Val	0.9929824561403509	True	3.4212456618605007e-06
84	580	p.Leu89Phe	0.9929824561403509	True	5.3293871080886465e-05
85	582	p.Ser90Gly	0.7789473684210526	False	3.6010082823190495e-06
86	583	p.Ser90Asn	0.7789473684210526	False	6.842266596601857e-07
87	584	p.Ser90Thr	0.7789473684210526	False	1.3684533193203714e-06
88	586	p.Ile91Met	0.968421052631579	True	2.7368317341113233e-06
89	587	p.Ser92Thr	0.9894736842105264	True	6.841938896012108e-07

Fig.25 Classification result of second obtained file

51	398	p.Tyr51His	0.7395833333333334	False	2.0526350360169026e-06
52	399	p.Pro52Leu	0.7395833333333334	False	1.2003274493281767e-06
53	401	p.Ala54Thr	0.7395833333333334	False	6.841892084204535e-07
54	402	p.Asn55Asp	0.7395833333333334	False	1.368335351627977e-06
55	403	p.Asn55Lys	0.7395833333333334	False	1.5909835778675093e-06
56	406	p.Leu56Val	0.7395833333333334	False	6.841620588352004e-07
57	410	p.Leu58Gln	0.7395833333333334	False	2.561449165479862e-06
58	412	p.Val60Met	0.7291666666666666	False	1.5909127066197878e-06
59	418	p.Leu63Phe	1.0	True	3.1818254132395756e-06
60	419	p.Leu63Pro	1.0	True	1.3683016503086204e-06
61	420	p.Leu64Phe	0.7708333333333334	False	3.42072136172076e-06
62	424	p.Leu65Phe	0.9583333333333334	True	6.84143336238662e-07
63	425	p.Leu65Val	0.9583333333333334	True	1.1772695278033886e-05
64	426	p.Leu65His	0.9583333333333334	True	1.5909127066197878e-06

Fig.26 Classification result of third obtained file

Results of t-test which were used for finding significant difference in allele frequencies between variants classified as pathogenic and nonpathogenic as p-values for three different data were shown in **Figure 27**.

```
P-value for classification_output.csv: 0.8108137642475657
P-value for classification_output2.csv: 0.6094924481573922
P-value for classification_output3.csv: 0.6531055759790125
```

Fig.27 p-values as a result of t-test

Lastly, conservation scores and allele frequency scatter plot of mutations were shown in **Figure 28, 29** and **30**.

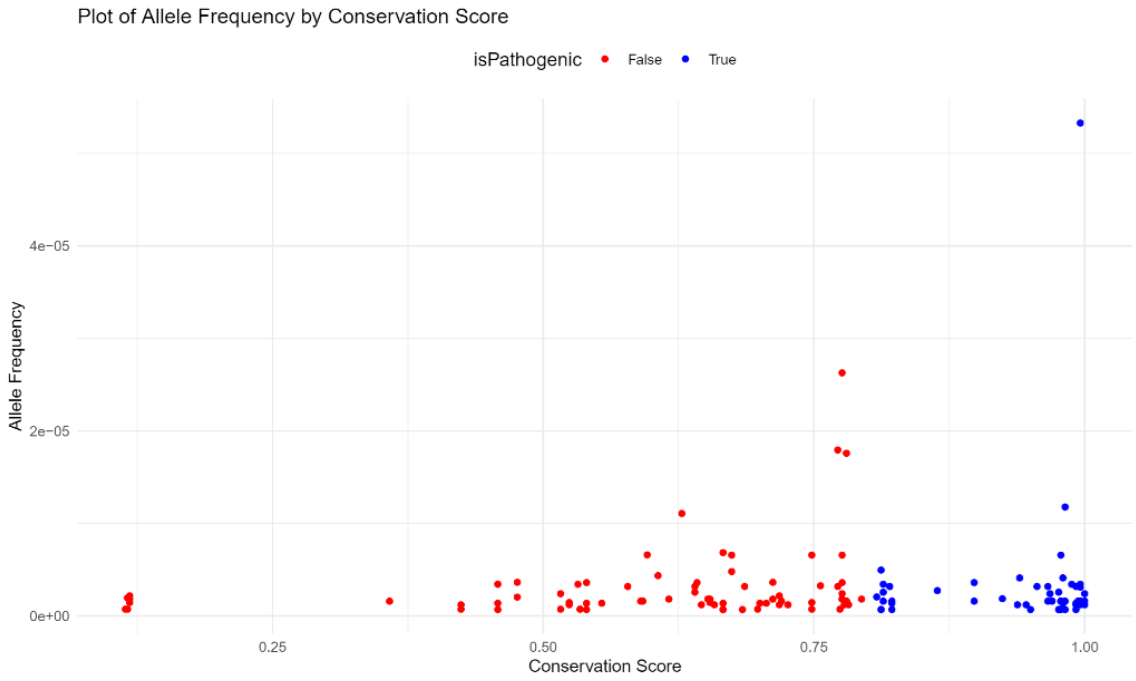

Fig.28 Conservation Scores and Allele Frequency scatter plot of mutations for 500 hits data

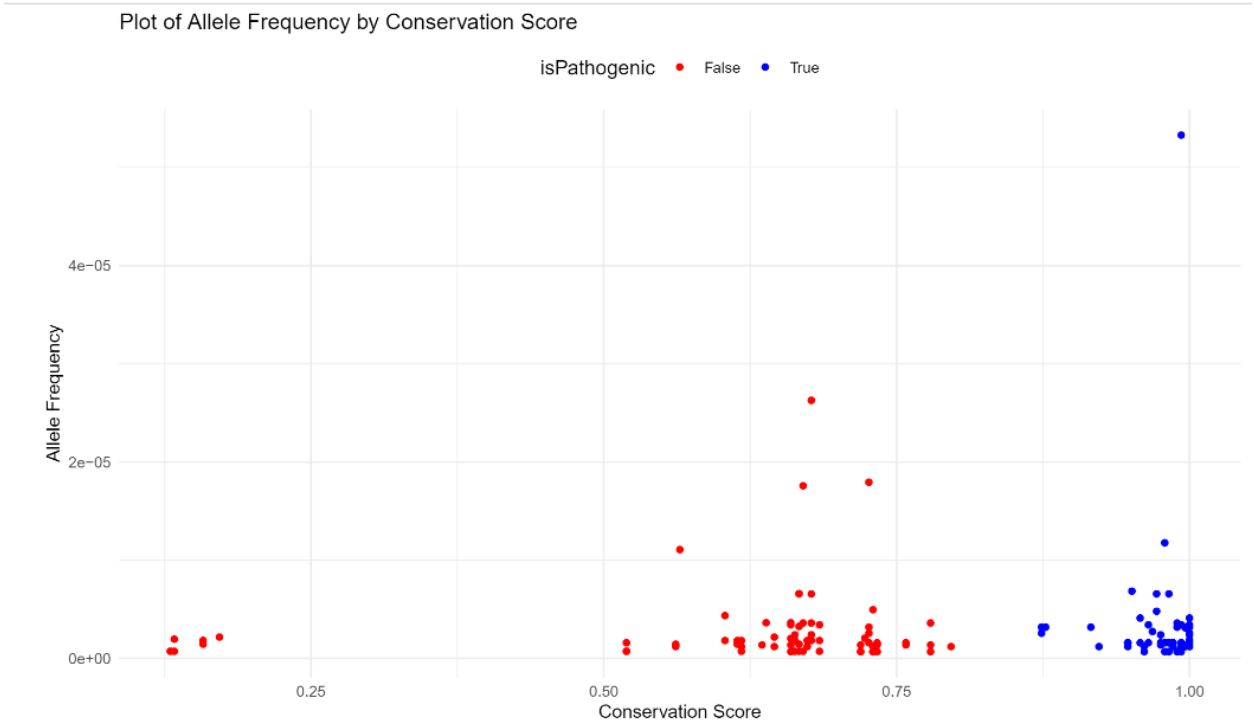

Fig.29 Conservation Scores and Allele Frequency scatter plot of mutations for subtree data

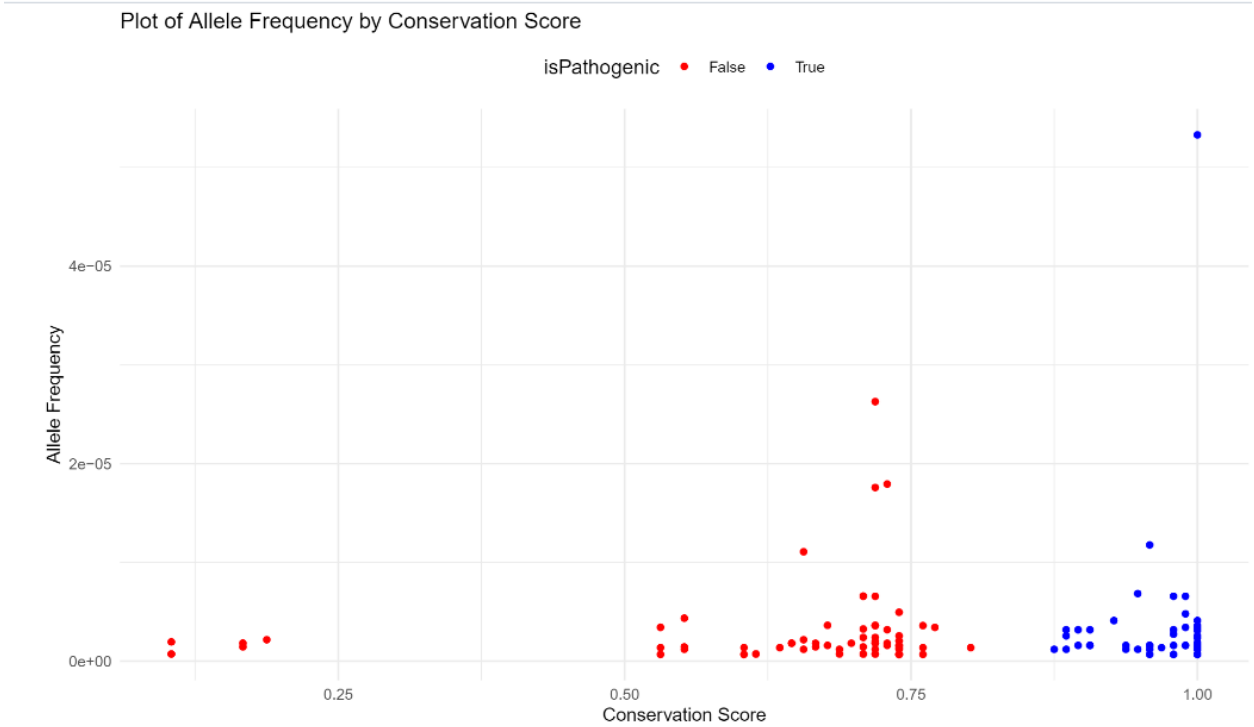

Fig.30 Conservation Scores and Allele Frequency scatter plot of mutations for pruned subtree data

D. DISCUSSION

The main idea of the project was making an analysis on how conserved the amino acids per position in the evolutionary process for a specific protein, TMEM216, to then determine the unknown variants as pathogenic and non pathogenic. Second purpose was to observe a relationship between allele frequencies of variants classified as pathogenic and nonpathogenic, and determining if the p-values are significant. For these purposes, data retrieved from BLASTp as 100, 250, 500, 1000, and 5000 hits of sequences. However, due to small coverage of 100 and 250 hits, and presence of a paralogous gene for Homo Sapiens in 5000 hits, these data are not used for the analysis. Also, 1000 hits data was not used because of the time consuming of the tree forming with this data, because of the speed of the devices and programs. In the end, 500 hits data was decided on to perform the purposes of the project.

In the alignment of the data with MEGAX tool MUSCLE algorithm, MUSCLE is chosen as the algorithm between ClustalW and MUSCLE algorithm. Both of the algorithms use a progressive alignment approach. However, because ClustalW uses a guide tree derived from pairwise distance, and MUSCLE is faster and more efficient, MUSCLE is chosen as the algorithm. After the alignment in MEGA, for the tree formation in the same tool, the Neighbor-Joining method is used. This method was chosen among Maximum-Likelihood method, Minimum-Evolution method, UPGMA method, Bayesian Inference method, and Neighbor-Joining method. The Maximum-Likelihood method was a better choice for the purpose of the project, but the Neighbor-Joining method is chosen because it provides a flexibility to construct multiple trees in a short time. For the forming a phylogenetic tree with Neighbor-Joining method, Bootstrap method is chosen as Phylogeny Test method, and 100 is used as default from the tool as a number of Bootstrap replications. For the Substitution Model, Amino Acid is used as Substitution Type, and Jones-Taylor-Thorton (JTT) model is used. Jones-Taylor-Thorton (JTT) model chosen between Dayhoff-Jones-Taylor (DJT) model and Jones-Taylor-Thorton (JTT) model. Since the JTT model is generally more complex than the Dayhoff model, amino acid substitution patterns can be represented in more depth, and choices were made according to this. For the Rates and Patterns, Gamma Distributed is used. It is chosen between Gamma Distributed and Uniform Rates. According to the Uniform Rates, each site in an alignment of sequences evolves at the same rate. Put differently, it presupposes a steady rate of substitution at every alignment location. Because of this reason, Gamma Distributed is chosen.

After the tree formation with MEGAX, Midpoint rerooting applied to the tree in Figtree tool. Midpoint rerooting is chosen between Outgroup rerooting and Midpoint rerooting. It is because it cannot be decided on the Outgroup in the tree for rerooting. Later, the clade which includes all Homo Sapiens isoforms is chosen to form the subtree. In this subtree, further

specification aimed and formation of the new tree with realignment and Midpoint rerooting applied again to this subtree. Also, Bootstrap values are used to filter the tree more. For this filtering, 0.7 is chosen as the threshold value. 0.7 is chosen according to the literature (Hillis & Bull, 1993). Species with Bootstrap values lower than 0.7 are eliminated from the data because of the low confidence level, and species with Bootstrap values bigger than 0.7 taken for the formation of new pruned subtree. If the Bootstrap value is determined higher than 0.7, more confidence level can be caught in the obtaining of the new subtree.

After the creation of the mentioned two data with the tree formation, and with the first 500 hit data, conservation scores and consensus sequences are determined. With the help of these conservation scores and consensus sequences, and the information of known pathogenic variants from the gnomAD's ClinVar data information part with the variants found from the papers, classification applied as pathogenic and nonpathogenic to the three different data. 13 known mutations, Arg12Cys, Arg12His, Arg12Leu, Gly16Ala, Leu53Arg, Arg73Leu, Arg85Ter, Arg73His, Gly77Ala, Thr78LysfsTer30, Leu133Ter, Arg73Cys, Leu114Arg are used in this manner. Due to the narrow scope of the literature on this topic, no more than 13 variants were found, and if there were more known variants, analysis of the project could be in higher accuracy. In the classification, threshold values computed with a Python code as 0.8074285714285713 for the first 500 hits data, 0.8401002506265665 for the subtree data, and 0.8705357142857143 for the pruned subtree data used. In the 500 hits data, because it has more gap insertions compared to the other two data, variant positions coincide with more gaps, and the score becomes low compared to the other two data. As the last step of the project, p-values which indicate the difference between the allele frequencies of variants classified as pathogenic and nonpathogenic are obtained. p-value for the first 500 hits data is 0.8108137642475657, p-value for the subtree data is 0.6094924481573922, and p-value for the pruned subtree data is 0.6531055759790125. Three of them greater than 0.05 indicates that there is no sufficient evidence to conclude that there is a statistically significant difference in allele frequencies between pathogenic and nonpathogenic variants. However, according to the conservation scores and allele frequency scatter plot of mutations can be seen in **Figure 28, 29**, and **30**, there is no negative correlation between conservation scores and allele frequencies.

E. REFERENCES

Agnihotry, S., Pathak, R. K., Singh, D. B., Tiwari, A., & Hussain, I. (2022). Protein structure prediction. *Bioinformatics*, 177–188. https://doi.org/10.1016/b978-0-323-89775-4.00023-7

ClinVar. (n.d.). *TMEM216[gene] - ClinVar - NCBI*. https://www.ncbi.nlm.nih.gov/clinvar/?term=TMEM216%5Bgene%5D&redir=gene

Edvardson, S., Shaag, A., Zenvirt, S., Erlich, Y., Hannon, G. J., Shanske, A., Gomori, J. M., Ekstein, J., & Elpeleg, O. (2010c). Joubert Syndrome 2 (JBTS2) in Ashkenazi Jews Is Associated with a TMEM216 Mutation. *The American Journal of Human Genetics*, *86*(2), 294. https://doi.org/10.1016/j.ajhg.2010.01.022

FigTree. (n.d.). http://tree.bio.ed.ac.uk/software/figtree/

GnomAD. (n.d.). *gnomAD Browser*. https://gnomad.broadinstitute.org/gene/ENSG00000187049?dataset=gnomad_r4

Hillis, D. M., & Bull, J. J. (1993). An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis. *Systematic Biology*, *42*(2), 182–192. https://doi.org/10.1093/sysbio/42.2.182

Koichiro Tamura, Glen Stecher, and Sudhir Kumar (2020) MEGAX: Molecular Evolutionary Genetics Analysis version X. *Molecular Biology and Evolution* 38:3022-3027. https://www.megasoftware.net/

OMIM. *Transmembrane protein 216; TMEM216*. (n.d.).

https://www.omim.org/entry/613277

Parisi, M. (2017, June 29). *Joubert Syndrome*. GeneReviews - NCBI Bookshelf. https://www.ncbi.nlm.nih.gov/books/NBK1325/

Protein BLAST: search protein databases using a protein query. (n.d.). https://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastp&PAGE_TYPE=BlastSearch&LINK_LOC=blasthome

UniProt. (n.d.). https://www.uniprot.org/

U.S. National Library of Medicine. (2017, July 1). *Joubert syndrome*. MedlinePlus. https://medlineplus.gov/genetics/condition/joubert-syndrome/#:~:text=Joubert%20syndrome%20typically%20has%20an,and%20symptoms%20of%20the%20condition

U.S. National Library of Medicine. (n.d.). *Joubert syndrome 2* . National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/medgen/334114

U.S. National Library of Medicine. (n.d.). *TMEM216 transmembrane protein 216 [homo sapiens (human)]* . National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/gene/51259

U.S. National Library of Medicine. (2021, March 26). *What are proteins and what do They do?*. MedlinePlus. https://medlineplus.gov/genetics/understanding/howgeneswork/protein/

Valente, E. M. et.al.(2010). Mutations in TMEM216 perturb ciliogenesis and cause Joubert, Meckel and related syndromes. *Nature Genetics*, *42*(7), 619–625. https://doi.org/10.1038/ng.594