Background

In recent years, gene therapy has become an increasingly attractive treatment for many genetic diseases, the greatest of these being cancer. With that in mind, the need for understanding the genes linked to cancer and its causes has never been greater.

The TP53 gene, often nicknamed the guardian of the genome, is an essential part of the body's tool set against cancer. This gene is known as a tumor suppressor gene. TP53 contains the instructions for a protein called p53 which attaches to DNA. From there, it detects damage in the DNA and responds accordingly to whether the DNA can be repaired. If the DNA is deemed repairable, it activates other genes to fix said damage; however, if the DNA is deemed irreparable, the protein prevents the cell from dividing and signals the cell to undergo apoptosis, or controlled cell death. In doing this, the protein prevents the formation of tumors, hence the name "guardian of the genome."

Mutations of this gene have been found to be responsible for many different types of cancers in humans, including but not limited to breast cancer, lung cancer, bladder cancer, as well as genetic syndromes like Li-Fraumeni Syndrome; however, this gene is not exclusively human. Other animals such as axolotl and lizards, animals with regenerative capabilities and cancer resistances, also contain this gene. By seeing the differences in their genes, we may be able to ascertain the origins in their resistance to cancer and hopefully one day find an application for them through gene therapy.

Prior work into the TP53 and the p53 protein have shown some promise into TP53 and p53's role in cancer resistance other animals. A study found that controlled regulation of p53 is essential for the regeneration of limbs in salamander and fish species (Yun 17392). Another

study found that the axolotl tp53 gene only had a 38 animo acid difference and suggested that techniques like CRISPR could be used in tandem with axolotl TP53 genes for gene therapy (Ahsen 30).

In this paper, we look to gather sequences of the TP53 gene of different animals from the GenBank database. Some of these animals will be random, others from animals with documented cancer resistant or regenerative traits such as axolotl and salamanders. We will then align these sequences and create a phylogenetic tree. From this tree, we can ascertain the evolutionary relationships between animals in terms of their TP53 gene, then compare what their shared traits, protein structures, and how this relates to their respective TP53 sequences.

From this process, we expect to see clustering between animals within the same category, such as animals within the order primates clustering. Additionally, we hope to see clustering between animals with shared traits like cancer resistance as we believe that animals with longer lifespans have more developed/complex TP53 genes to counteract the susceptibility to diseases and viruses that comes with age, resulting in enhanced cancer resistance.

Methods

For our study, the data was obtained from the National Center for Biotechnology Information (NCBI), a popular and reliable resource for genetic and biomedical related information. They provide public access to large genome databases, including sequences from GenBank, RefSeq, and other sources. Our project's data relied on NCBI's Tumor Protein P53 orthologs and nucleotide databases to collect relevant sequences. These orthologs are a collection of genes in different species, evolving from a common ancestor. And in our case, they were used for analyzing evolutionary relationships and gene conservation across species. Using these

databases, we retrieved wild-type TP53 sequences from both mammalian and non-mammalian species, focusing on the coding regions to examine nucleotide variations.

We also included two disease-associated sequences. Squamous cell carcinoma, a type of skin cancer, is represented by a TP53 sequence from a horse (RefSeq: NM_001202405). This sequence allows us to explore cancer development in other genomic regions outside of the TP53 and its potential effects on the TP53 gene. We also included a TP53 sequence containing the woodchuck hepatitis virus, a virus associated with liver disease and cancer in woodchucks (RefSeq: AJ001022.1), giving us the ability to study the interaction between viral infections and tumor-suppressing mechanisms. This comparison may prove to us its significance, as it allows us to investigate whether infectious changes are localized or reflect greater evolutionary pressures. All collected sequences were organized into two separate .fasta files for analysis: one for human and mammalian sequences and one for human and non-mammalian sequences.

Our sequences were extracted as nucleotides to help identify any nucleotide insertions or deletions hidden by silent mutations. But for ease of analysis and reading, we translated our nucleotides into amino acids, through a Python program. Within the program, a codon table was implemented using Python dictionaries, mapping nucleotide triplets to their corresponding amino acids. The program then parses each sequence, reading codons in multiples of 3 and appending the corresponding amino acid to a new sequence. The final translated sequences were saved into new .fasta and cross-checked against Bioinformatics.org's Sequence Manipulation Suite to ensure accuracy. These nucleotide and amino acid sequences were then subjected to alignment algorithms.

To help identify any patterns of gene conservation and possible evolutionary divergence, we utilized multiple sequence alignment (MSA) algorithms, which are helpful for comparing biological sequences. MSA's are used to align three or more biological sequences, as opposed to pairwise alignments, to identify regions of similarity. These regions help all researchers identify evolutionary patterns or functions. To perform a MSA, we implemented our own version of the Needleman-Wunsch alignment algorithm using Python. Needleman-Wunsch aligns sequences calculating the maximum similarity score across their entire lengths, making it an ideal method for studying conserved genes like TP53. The scoring system used was: Match: +1; Mismatch: -1; and Gap/Indel: -2.

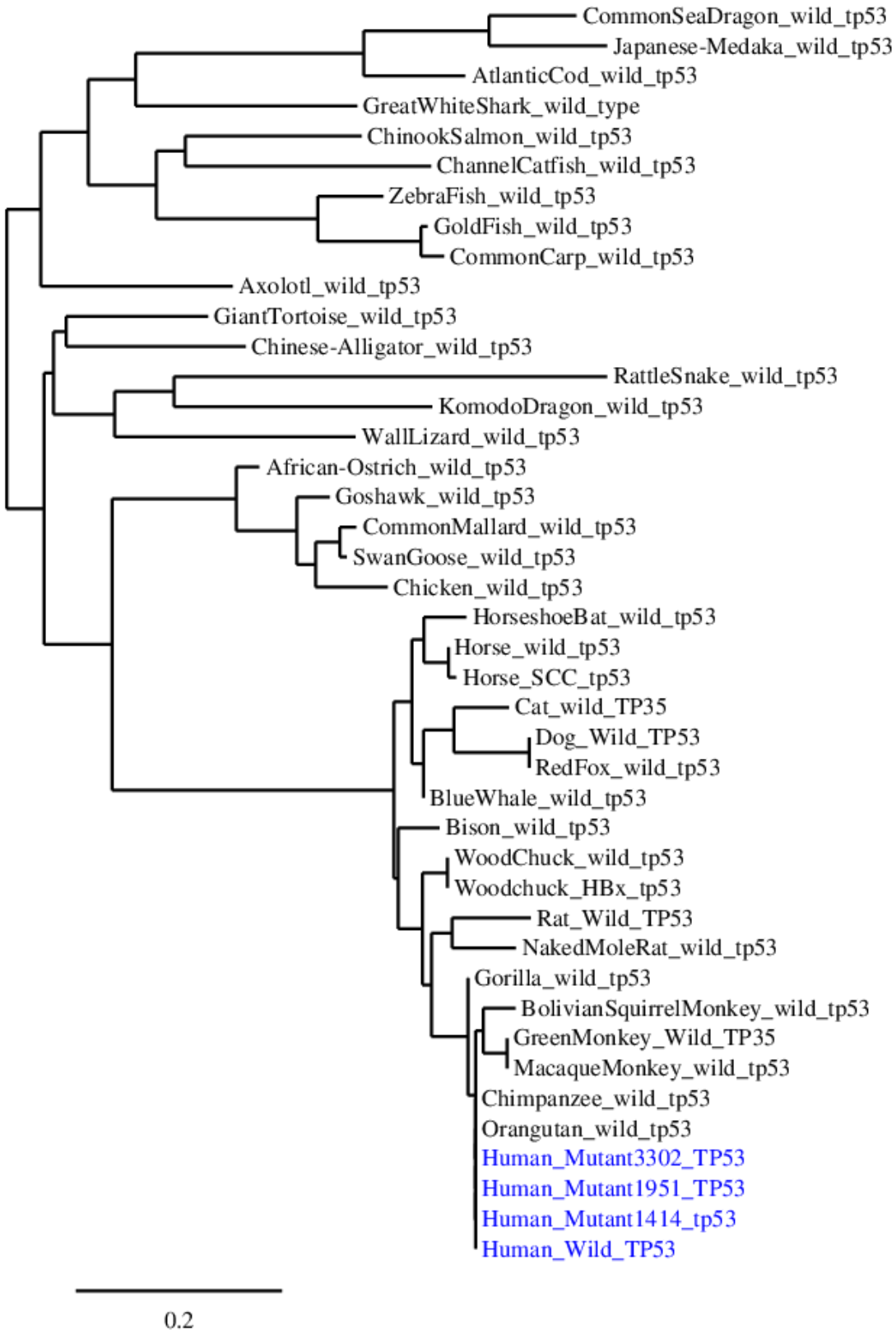
The algorithm calculates a matrix, as a 2d array, filling it iteratively with scores based on sequence comparisons. We have also implemented a traceback function to then retrieve the optimal alignment path, ensuring a strong match between sequences. By focusing on global alignment, Needleman-Wunsch ensures that even the ends of sequences are considered, which is critical for full-length gene comparisons. Several additional functions were incorporated as part of our implementation of a MSA: The Alignment Matrix Construction function calculated scores for each cell in the alignment matrix based on any matches, mismatches, and gaps. Our Traceback Path function backtracks through the entire matrix to search for the alignment with the maximum score. Additionally, the Consensus Sequence Calculation function identifies the most common nucleotide or amino acid at each column of the alignment. This highlights conserved regions that are functionally or structurally significant. Finally, the Gap Refinement function analyzes and removes trailing or unnecessary gaps to improve alignment readability.

Since our custom algorithm is a simple representation of a MSA tool, we needed to validate and supplement our results with the help of Clustal Omega, a popular bioinformatic tool

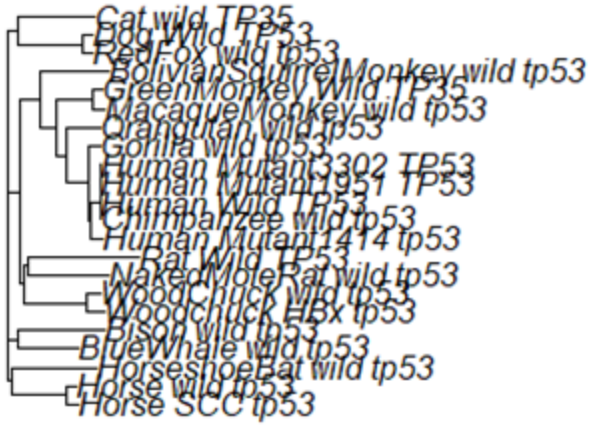
developed by the European Molecular Biology Laboratory. Clustal Omega uses much more advanced strategies, such as progressive alignments, to handle large and complex alignments with greater precision. Although our approach utilized Needleman Wunsch's pairwise algorithm and progressive alignments, similar to Clustal Omega, we needed to strengthen the reliability of our own results. Comparing our own MSA tool with those generated by Clustal Omega helps give us accurate results, providing a strong basis for constructing phylogenetic trees to further explore evolutionary relationships.

Phylogenetic trees are used to visually depict evolutionary relationships between species by organizing sequences and their genetic similarities. Analyzing our phylogenetic trees helped us understand how the TP53 gene has evolved across different taxa. We constructed trees based on the MSA results using our own R program, which integrates several bioinformatics related tools. The program aligns sequences using the msa package with Clustal Omega, calculates a distance matrix using a seqinr package, and builds the phylogenetic tree using a neighbor-joining algorithm provided by the ape package. This method helps us ensure an accurate representation of evolutionary distances by clustering sequences based on their similarities. The resulting tree was plotted using base R, providing a clear visualization of TP53 gene relationships. To verify the accuracy of our constructed trees, we also used Phylogeny.fr, a popular bioinformatics tool used for reliable phylogenetic analyses by integrating multiple programs into a single streamlined workflow. Uploading our fasta files onto the web-based tool, the results were then compared with our R program's outputs. To help strengthen the accuracy of our results.

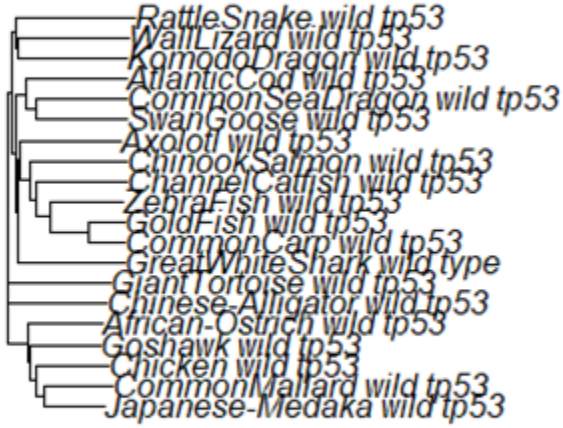
Results


CommonSeaDragon_wild_tp53
Japanese-Medaka_wild_tp53
AtlanticCod_wild_tp53
GreatWhiteShark_wild_type
ChinookSalmon_wild_tp53
ChannelCatfish_wild_tp53
ZebraFish_wild_tp53
GoldFish_wild_tp53
CommonCarp_wild_tp53
Axolotl_wild_tp53
GiantTortoise_wild_tp53
Chinese-Alligator_wild_tp53
RattleSnake_wild_tp53
KomodoDragon_wild_tp53
WallLizard_wild_tp53
African-Ostrich_wild_tp53
Goshawk_wild_tp53
CommonMallard_wild_tp53
SwanGoose_wild_tp53
Chicken_wild_tp53
HorseshoeBat_wild_tp53
Horse_wild_tp53
Horse_SCC_tp53
Cat_wild_TP35
Dog_Wild_TP53
RedFox_wild_tp53
BlueWhale_wild_tp53
Bison_wild_tp53
WoodChuck_wild_tp53
Woodchuck_HBx_tp53
Rat_Wild_TP53
NakedMoleRat_wild_tp53
Gorilla_wild_tp53
BolivianSquirrelMonkey_wild_tp53
GreenMonkey_Wild_TP35
MacaqueMonkey_wild_tp53
Chimpanzee_wild_tp53
Orangutan_wild_tp53
Human_Mutant3302_TP53
Human_Mutant1951_TP53
Human_Mutant1414_tp53
Human_Wild_TP53
0.2

Mammal Phylogenetic Tree



Non-Mammal Phylogenetic Tree



Discussion

To investigate whether longer living organisms are more likely to have more complex TP53 genes resulting in higher cancer resistance, the sequences gathered were used to produce multiple sequence alignments, the process of which is detailed in the methods section of the paper. These alignments were then used to create phylogenetic trees through two methods. One tree was created with software written for this study, implemented in R using bioinformatic based packages. In short, this software utilized the alignments produced to calculate a distance matrix, then used that distance matrix to generate a phylogenetic tree through an R implementation of Clustral Omega. The other tree created was from a third-party program, also utilizing Clustral Omega as mentioned in the methods section of this paper.

From this process, we found that the mammalian portion of sequences produced a multiple sequence alignment that showed much less variance in the TP53 sequences when compared to the multiple sequence alignment produced by the non-mammalian portion of sequences. This was measured by the overall presence of indel or gaps in the multiple sequence alignments as well as the distance values calculated from the R implementation. This result is in line with what was expected as the mammalian portion of sequences were curated all from the same order of organism, while the non-mammalian portion of sequences were not exclusively from one order. Additionally, both portions of sequences also included subjects with mutations and diseases that could affect the TP53 gene or be affected by defects in the TP53 gene, further promoting the variance that was observed in the multiple sequence alignments.

The phylogenetic trees produced offered different results. The collective tree, containing all of the sequences gathered regardless of category, as well as the non-mammalian tree showed that organisms that exhibited the desirable traits of cancer resistance and regenerative capabilities

were unique and didn't offer any correlation to age or to evolutionary relation. The axolotl, one of the organisms that exhibited these traits, acted as an out group in the non-mammalian phylogenetic tree and was isolated with its clade in the collective tree. Additionally, when looking at age, axolotls have a life span of 5 years in the wild despite possessing such desirable traits. Alone, this suggested that age was not a factor in developing cancer resistance.

However, considering the many factors that affect evolution, as well as the expected life span of an organism, it's clear to see how no significant connection between age and TP53 complexity was found. A multiple sequence alignment can only show how similar two sequences are, and phylogenetic tree can only give rough evolutionary relationships between sequences. If this study were to be repeated, it would be helpful to ascertain the environments where the organisms lived to get an idea of environmental pressures place on the organisms. It would also help to classify organisms, minimizing their effect as confounding variables.

Hypothetically, given an infinite budget to study this question, the first change to this study would be the long-term gathering of data to study the development of cancers in certain environments. From that data, TP53 sequences could be gathered from proportions of organism populations that each do and do not have each type of cancer. Categorized by environment and relative age, this would isolate factors for cancer resistance down to individual experience and gene makeup, showing what mutations cause and create resistance to cancer. This data could then be compared across organisms of differing life-spans, but similar environments to ascertain whether life-span is a major identifier of cancer resistance and other such traits.

References

Ahsen Babar, Hale, et al. “Expression of axolotl p53 gene may increase cancer resistance.” Gene Editing, vol. 3, no. 3, 21 Dec. 2022, pp. 30–34, https://doi.org/10.29228/genediting.66929.

Institute, European Bioinformatics. “EMBL-Ebi Homepage.” Homepage, www.ebi.ac.uk/. Accessed 25 Nov. 2024.

Temogoh, Bryan. “How Can I Generate a Phylogenetic Tree in R Starting from a FASTA File?” Applied Epi Community, 28 Apr. 2022, community.appliedepi.org/t/how-can-i-generate-a-phylogenetic-tree-in-r-starting-from-a-fasta-file/221.

Yun, Maximina H., et al. “Regulation of p53 is critical for vertebrate limb regeneration.” Proceedings of the National Academy of Sciences, vol. 110, no. 43, 7 Oct. 2013, pp. 17392–17397, https://doi.org/10.1073/pnas.1310519110.

“ÉQUIPE MAB : Méthodes et Algorithmes Pour La Bio Informatique.” LIRMM, www.lirmm.fr/equipes/mab/. Accessed 25 Nov. 2024.

Appendices

Phylogenetic Tree in Newich Format:

format:((((((((((Human_Wild_TP53:0.0,(Human_Mutant1414_tp53:0.0,(Human_Mutant1951_TP53:0.0,(Human_Mutant3302_TP53:0.0,Orangutan_wild_tp53:0.0):0.0):0.0):0.0):0.0):0.0,Chimpanzee_wild_tp53:0.0):0.0,((MacaqueMonkey_wild_tp53:0.0,GreenMonkey_Wild_TP35:0.0):0.0233,BolivianSquirrelMonkey_wild_tp53:0.03114):0.00736):0.00736,Gorilla_wild_tp53:0.0001):0.03558,(NakedMoleRat_wild_tp53:0.06185,Rat_Wild_TP53:0.0762):0.01993):0.00873,(Woodchuc

k_HBx_tp53:0.0,WoodChuck_wild_tp53:0.0):0.02421):0.02341,Bison_wild_tp53:0.03983):0.00434,((BlueWhale_wild_tp53:0.0,((RedFox_wild_tp53:0.0,Dog_Wild_TP53:0.0):0.07298,Cat_wild_TP35:0.0535):0.02955):0.01134,((Horse_SCC_tp53:0.0076,Horse_wild_tp53:0.0):0.02384,HorseshoeBat_wild_tp53:0.04119):0.01165):0.01737):0.27340,(((Chicken_wild_tp53:0.06994,(SwanGoose_wild_tp53:0.00684,CommonMallard_wild_tp53:0.01588):0.02349):0.01817,Goshawk_wild_tp53:0.03174):0.05867,African-Ostrich_wild_tp53:0.02223):0.1206):0.066,((WallLizard_wild_tp53:0.2336,(KomodoDragon_wild_tp53:0.2504,RattleSnake_wild_tp53:0.4202):0.05756):0.05953,(Chinese-Alligator_wild_tp53:0.1742,GiantTortoise_wild_tp53:0.1378):0.01241):0.00901):0.03615,(Axolotl_wild_tp53:0.18610,((((CommonCarp_wild_tp53:0.02252,GoldFish_wild_tp53:0.00643):0.1,ZebraFish_wild_tp53:0.06368):0.1569,(ChannelCatfish_wild_tp53:0.2385,ChinookSalmon_wild_tp53:0.1711):0.02815):0.06582,(GreatWhiteShark_wild_type:0.215,(AtlanticCod_wild_tp53:0.09853,(Japanese-Medaka_wild_tp53:0.1144,CommonSeaDragon_wild_tp53:0.08462):0.1218):0.2211):0.04607):0.04614):0.03293);

MSA - MAMMALS

>Rat_Wild_TP53

MEDSQSDMSIELPLSQETFSCLWKLLPPDDILPTTATGSPNSMEDLFL-PQDVAELLEG---PEEALQVSA-PAAQEPGTEAPAPVAPASATPWPLSSSVPSQKTYQGNYGFHLGFLQSGTAKSVMCTYSISLNKLFCQLAKTCPVQLWVTSTPPPGTRVRAMAIYKKSQHMTEVVRRCPHHER--------CSDGDGLAPPQHLIRVEGNPYAEYLDDRQTFRHSVVVPYEPPEVGSDYTTIHYKYMCNSSCMGGMNRRPILTIITLEDSSGNLLGRDSFEVRVCACPGRDRRTEEENFRKKEEHCPELPPGSAKR----------AL
PTSTSSSPQQKKKPLDGEYFTLKIRGRERFEMFRELNEALELKDARAAEESGDSRAHSSYPKTKKGQSTSRHKKPMIKKVGPDSD

>BolivianSquirrelMonkey_wild_tp53

MEEPQSDLSIEPPLSQETFSDLWKLLPENNILSSSLSQ---PVDDLMLSPDDIAQWFSQDPVPDESPTVSEAPPAVAQAPAAPTSAAPTPAPSWPLSSSVPSQKTYHGDYGFRLGFLHSGTAKSVTCTYSPALNKMFCQLAKTCPVQLWVDSTPPRGTRVRAMAIYKQSQHMTEVVRRCPHH--------ERCSDSDGLAPPQHLIRVEGNLHVEYLDDKNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTIITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENFRKKGEPCLDLPPGSTKR----------

AMPNSTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKS
KKGQCTSRHKKLMVKREGPDSD

>GreenMonkey_Wild_TP35

MEEPQSDPSIEPPLSQETFSDLWKLLPENNVLSPLPSQ---AVDDLMLSPDDLAQWLTEDPGPDEAP
RMSEAAPPMAPTPAAPTPAAPAPAPSWPLSSSVPSQKTYHGSYGFRLGFLHSGTAKSVTCTYSPDL
NKMFCQLAKTCPVQLWVDSTPPPGSRVRAMAIYKQSQHMTEVVRRCPHH--------ERCSDSDGLA
PPQHLIRVEGNLRVEYSDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTII
TLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENFRKKGEPCHELPPGSTKR----------
-ALPNNTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPAGSRAHSSHLKS
KKGQSTSRHKKFMFKTEGPDSD

>MacaqueMonkey_wild_tp53

MEEPQSDPSIEPPLSQETFSDLWKLLPENNVLSPLPSQ---AVDDLMLSPDDLAQWLTEDPGPDEAP
RMSEAAPPMAPTPAAPTPAAPAPAPSWPLSSSVPSQKTYHGSYGFRLGFLHSGTAKSVTCTYSPDL
NKMFCQLAKTCPVQLWVDSTPPPGSRVRAMAIYKQSQHMTEVVRRCPHH--------ERCSDSDGLA
PPQHLIRVEGNLRVEYSDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTII
TLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENFRKKGEPCHQLPPGSTKR----------
-ALPNNTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPAGSRAHSSHLKS
KKGQSTSRHKKFMFKTEGPDSD

>Orangutan_wild_tp53

MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQ---AVDDLLLSPDDIAQWFIEDPGPDEAPR
MSEAASPVDPAPAAPIPAAPAPAPSWPLSSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPALN
KMFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHH--------ERCSDSDGLAP
PQHLIRVEGNLRVEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTIIT
LEDSSGNLLGRNSFEVRVCACPGRDRRTEEENFRKKGEPHHELPPGSTKR----------
ALPNNTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKS
KKGQSTSRHKKLMFKTEGPDSD

>Human_Mutant3302_TP53

MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQ---AMDDLMLSPDDIEQWFTEDPGPDEAP
RMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPAL
NKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHH--------ERCSDSDGLA
PPQHLIRVEGNLRVEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPIL----
-EDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKGEPHHELPPGSTKR----------A
LPNNTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKSK
KGQSTSRHKKLMFKTEGPDSD

>Human_Mutant1951_TP53

MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQ---AMDDLMLSPDDIEQWFTEDPGPDEAP
RMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPAL
NKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHHERCCCPHHERCSDS
DGLAPPQHLIRVEGNLRVEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRR

PILTIITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKGEPHHELPPGSTKR----------
---------ALPNNTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSS
HLKSKKGQSTSRHKKLMFKTEGPDSD

>Gorilla_wild_tp53

MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQ---AMDDLMLSPDDIEQWFTEDPGPDEAP
RMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPTL
NKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHH--------ERCSDSDGLA
PPQHLIRVEGNLRVEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTII
TLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKGEPHHELPPGSTKR----------
-ALPNNTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKS
KKGQSTSRHKKLMFKTEGPDSD

>Human_Wild_TP53

MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQ---AMDDLMLSPDDIEQWFTEDPGPDEAP
RMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPAL
NKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHH--------ERCSDSDGLA
PPQHLIRVEGNLRVEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTII
TLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKGEPHHELPPGSTKR----------
-ALPNNTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKS
KKGQSTSRHKKLMFKTEGPDSD

>Chimpanzee_wild_tp53

MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQ---AMDDLMLSPDDIEQWFTEDPGPDEAP
RMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPAL
NKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHH--------ERCSDSDGLA
PPQHLIRVEGNLRVEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTII
TLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKGEPHHELPPGSTKR----------
-ALPNNTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKS
KKGQSTSRHKKLMFKTEGPDSD

>Human_Mutant1414_tp53

MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQ---AMDDLMLSPDDIEQWFTEDPGPDEAP
RMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYGFRLGFLHSGTAKSVTCTYSPAL
NKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHH--------ERCSDSDGLA
PPQHLIRVEGNLRVEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTII
TLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKGEPHHELPPGSTKR----------
-ALPNNTSSSPQPKKKPLDGEYFTLQIRGREHFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKS
KKGQSTSRHKKLMFKTEGPDSD

>NakedMoleRat_wild_tp53

MEEPQSDLSIEPPLSQETFSDLWKLLPENNVLSSSLSS---PMDDLLLSPEDVVNWLGGN--PDEDVQ
VSAAPVPEPPTPVAPAPAAPAPATSWPLSSSVPSHKTYQGNYGFHLGFLQSGTAKSVTCTYSPVLN
KLFCQLAKTCPVQVWVESPPPPGTRVRAMAIYKKSQHMTEVVRRCPHH--------ERCSDSDGLAPP

QHLIRVEGNLRAEYLDDRTTFRHSVVVPYDLPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTIITL
EDSSGNLLGRNSFEVRVCACPGRDRRTEEENFHKKGGSCPEPTPGSIKR----------AL
PTGTNSSPQPKKKPLDGEYFTLKIRGRERFEMFRELNEALELKDAQTEKEPGESRPHSSYLKSKKG
QSTSCHKKLMFKKEGPDSD

>WoodChuck_wild_tp53

MEEPQSDLSIEPPLSQETFSDLWNLLPENNVLSPVLSP---PMDDLLLSSEDVENWFDKG--PDEALQ
MSAAPAPKAPTPAASTLAAPSPATSWPLSSSVPSQNTYPGVYGFRLGFLHSGTAKSVTCTYSPSLN
KLFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKKSQHMTEVVRRCPHH--------ERCSDSDGLAPP
QHLIRVEGNLRAEYLDDRNTFRHSVVVPYEPPEVGSECTTIHYNYMCNSSCMGGMNRRPILTIITL
EDSSGNLLGRNSFEVRVCACPGRDRRTEEENFRKRGEPCPEPPPRSTKR----------AL
PNGTSSSPQPKKKPLDGEYFTLKIRGRARFEMFQELNEALELKDAQAEKEPGESRPHPSYLKSKK
GQSTSRHKKIIFKREGPDSD

>Woodchuck_HBx_tp53

MEEAQSDLSIEPPLSQETFSDLWNLLPENNVLSPVLSP---PMDDLLLSSEDVENWFDKG--PDEALQ
MSAAPAPKAPTPAASTLAAPSPATSWPLSSSVPSQNTYPGVYGFRLGFLHSGTAKSVTCTYSPSLN
KLFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKKSQHMTEVVRRCPHH--------ERCSDSDGLAPP
QHLIRVEGNLRAEYLDDRNTFRHSVVVPYEPPEVGSECTTIHYNYMCNSSCMGGMNRRPILTIITL
EGSSGNLLGRNSFEVRVCACPGRDRRTEEENFRKRGEPCPEPPPRSTKR----------AL
PNGTSSSPQPKKKPLDGEYFTLKIRGRARFEMFQELNEALELKDAQAEKEPGESRPHPSYLKSKK
GQSTSRHKKIIFKREGPDSD

>HorseshoeBat_wild_tp53

MEVPQSELSVDPPLSQETFSDLWKLLPENNVLTPDV----------SLADLVNWLDGG--PNEDPNVPAT
PGPAAATPAP--ATSPAPANSWPLSSFVPSQKTYPGNYGFQLGFLNSGTAKSVTCTYSPTLNKLFCQ
LAKTCPVQLWVSSPPPVGTRVRAMAIYKKSEYMTEVVRRCPHHERCS-------DYSDGLAPPQHLIR
VEGNLHAEYLDDKHTFRHSVVVPYEPPEVGSDCTTIHYNFMCNSSCMGGMNRRPILTIITLEDSS
GNLLGRNSFEVRVCACPGRDRRTEEENFRKKGEPCPKQPPGSSKR----------ALPTN
TSSSTP-PKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKESEGSRAHSSHLKSKKGQST
SRHKKLLFKREGPDSD

>Cat_wild_TP35

MQEPPLELTIEPPLSQETFSELWNLLPENNVLSSELSS---AMNELPL-SEDVANWLDEA--PDDASGM
SAVPAPAAPAP-----ATPAPAISWPLSSFVPSQKTYPGAYGFHLGFLQSGTAKSVTCTYSPPLNKLFCQ
LAKTCPVQLWVRSPPPPGTCVRAMAIYKKSEFMTEVVRRCPHHERCP-------DSSDGLAPPQHLIR
VEGNLHAKYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNFMCNSSCMGGMNRRPIITIITLEDSN
GKLLGRNSFEVRVCACPGRDRRTEEENFRKKGEPCPEPPPGSTKRGKRAGREEAGRVQFGSKFTL
LSPFLTSFPALPPSTSSTPPQKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQSGKEPGGSRA
HSSHLKAKKGQSTSRHKKPMLKREGLDSD

>Dog_Wild_TP53

MQEPQSELNIDPPLSQETFSELWNLLPENNVLSSELCP---AVDELLL-PESVVNWLDED--SDDAPR
MPATSAPT----------APGPAPSWPLSSSVPSPKTYPGTYGFRLGFLHSGTAKSVTWTYSPLLNKLFCQ

LAKTCPVQLWVSSPPPPNTCVRAMAIYKKSEFVTEVVRRCPHHERCS-------DSSDGLAPPQHLIRV
EGNLRAKYLDDRNTFRHSVVVPYEPPEVGSDYTTIHYNYMCNSSCMGGMNRRPILTIITLEDSSG
NVLGRNSFEVRVCACPGRDRRTEEENFHKKGEPCPEPPPGSTKR----------ALPPSTS
SSPPQKKKPLDGEYFTLQIRGRERYEMFRNLNEALELKDAQSGKEPGGSRAHSSHLKAKKGQSTS
RHKKLMFKREGPDSD

>RedFox_wild_tp53

MQEPQSELNIDPPLSQETFSELWNLLPENNVLSSELCP---AVDELLL-PESVVNWLDED--SDDAPR
MPATSAPT----------APGPAPSWPLSSFVPSPKTYPGTYGFRLGFLHSGTAKSVTWTYSPLLNKLFCQ
LAKTCPVQLWVSSPPPPNTCVRAMAIYKKSEFVTEVVRRCPHHERCS-------DSSDGLAPPQHLIRV
EGNLRAKYLDDRNTFRHSVVVPYEPPEVGSDYTTIHYNYMCNSSCMGGMNRRPILTIITLEDSSG
NVLGRNSFEVRVCACPGRDRRTEEENFHKKGEPCPEPPPGSTKR----------ALPPSTS
SSPPQKKKPLDGEYFTLQIRGRERYEMFRNLNEALELKDAQSGKEPGGSRAHSSHLKAKKGQSTS
RHKKLMFKREGPDSD

>Bison_wild_tp53

MEESQAELNVEPPLSQETFSDLWNLLPENNLLSSELSA---PVDDLL-PYTDVATWLDEC--PNEAPQ
MPEPSAPAAPPP-----ATPAPATSWPLSSFVPSQKTYPGNYGFRLGFLHSGTAKSVTCTYSPSLNKLF
CQLAKTCPVQLWVDSPPPPGTRVRAMAIYKKLEHMTEVVRRCPHHERSS-------DYSDGLAPPQH
LIRVEGNLRAEYLDDRNTFRHSVVVPYESPEIDSECTTIHYNFMCNSSCMGGMNRRPILTIITLEDS
CGNLLGRNSFEVRVCACPGRDRRTEEENLRKKGQSCPEPPPRSTKR----------ALPT
NTSSSPQPKKKPLDGEYFTLQIRGFKRYEMFRELNDALELKDALDGREPGESRAHSSHLKSKKRP
SPSCHKKPMLKREGPDSD

>BlueWhale_wild_tp53

MEESQAELGVEPPLSQETFSDLWKLLPENNLLSSELSP---AVDDLLLSPEDVANWLDER--PDEAPQ
MPEPTAPAAPAP-----AAPAPATSWPLSSFVPSQKTYPGSYGFRLGFLHSGTAKSVTCTYSPALNKLF
CQLAKTCPVQLWVSSPPPPGTRVRAMAIYKKSEYMTEVVRRCPHHERCS-------DYSDGLAPPQH
LIRVEGNLRAEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNFMCNSSCMGGMNRRPILTIITLED
SGGNLLGRNSFEVRVCACPGRDRRTEEENFHKKGQSGPEPPPGSAKR----------ALP
TSTSSSPPQKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGESRAHSSHLKSKKG
QSPSRHKKLMFKREGPDSD

>Horse_wild_tp53

MEETQTELGIEPPLSQETFSDLWKLLPENNVLSPDLSP---AVNNLLLSP-DVVNWLDEG--PDEAPR
MP-----AAPAP-----LAPAPATSWPLSSFVPSQKTYPGCYGFRLGFLNSGTAKSVTCTYSPTLNKLFCQ
LAKTCPVQLLVSSPPPPGTRVRAMAIYKKSEFMTEVVRRCPHHERCS-------DSSDGLAPPQHLIRV
EGNLRAEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNFMCNSSCMGGMNRRPILTIITLEDSSGN
LLGRNSFEVRVCACPGRDRRTEEENFRKKEEPCPEPPPRSTKR----------VLSSNTSSS
PPQKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQTGKEPGGSKAHSSHLKSKKGQSTSSH
KKLIFKREGPDSD

>Horse_SCC_tp53MEETQTELGIEPPLSQETFSDLWKLLPENNVLSPDLSP---AVNNLLLSP-DVVNW
LDEG--PDEAPRMP-----AAPAP-----LAPAPATSWPLSSFVPSQKTYPGCYGFRLGFLNSGTAKSVTCT

YSPTLNKLFCQLAKTCPVQLLVSSPPPPGTRVRAMAIYKKSEFMTEVVRRCPHHERCS-------DSSD
GLAPPQHLIRVEGNLRAEYLEDRNTFRHSVVVPYEPPEVGSDCTTIHYNFMCNSSCMGGMNRRPI
LTIITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENFRKKEEPCPEPPPRSTKR----------
-----VLSSNTSSSPPQKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQTGKEPGGSKAHSSHL
KSKKGQSTSSHKKLIFKREGPDSD

MSA - NON MAMMALS

>WallLizard_wild_tp53

-------MEQLFESDAVFNELWTCMQSSKDFE---------EE-------LHKDLTL-------E--T----------
---FDLPFSEEG-SAP--QAGKHSSEATILPSTSAVPSTENYPGEHGFELAFEPSGTAKSVTCTY-PDLNK
LYCQMGKTCPVLIKVTTPPPPWAVIRTTAIFKKSEHIAEVVKRCPHHERSGEGSRGGIAPAEHLIRV
EGNLQAQYLSDQNTKRHSVVVPYEAPQLGTESSKILYNFMCNSSCQGGMNRRPILIIITLETHLGQ
LLGRRCFEARVCACPGRDRKTEEENFKKMAPSGINAKRNKLESDGGGNTKRSRLEAGS-GNAKQ
CKSEEESSQEWLPGSPGASFSQSSDSGGSKKRIVEAPSGVGSSTEGRVYTIQTRDRKLYQLLKKIK
EAIEFQDLMKQNKEPAKVLQKKM----------AVVVKDN----------

>AtlanticCod_wild_tp53

----------MDV
V--------EDEVLSLPLRQDSFQAFWESVPA---------TLDCLDPSPGSWFDL----------MPDS----------
----------SITV-EDLFP--------DPPAP-LDAPPTHAAPTVPTTSDYAGEYGFHIRFQNSGTAKSVTSTYS
VSLNKLFCQLAKTCPVEVLVDGAPPPGAILRATAVYKKTEHVADVVRRCPHHLK----E-DAVEHRS
HLIRVEGSQRAQYMEDPNTKRQSVTLPYEAPQLGSGHTTLLLNFMCNSSCMGGMNRRAILTILTL
ESSEGHVLGRRCFEVRVCACPGRDRKTEEGNVEKKTEGSKPTKK----------RKSPPTPATAAPPKRV
VS----------ASSAEEEDKEVFVLQVVGRKRYEVLKQINDALALQERMKVKQEV-
----------QGGPSRGKRRLGDRTDEGTD----------

>CommonSeaDragon_wild_tp53

----------MESS-
----LDDLSISQELQMSQESFLKVWGTYLTCPFPPNED------NDPSS--------TNQ----------LQGL----------
----------PEQFDENLFN--------MAAEAPVSDGVPPPASTVPVTTDHPGDYGFLLRFHNSGTAKSVTS
TYSVMLDKLYCQLAKTSPVEVLVSKEPPAGAVLRATAVYKKTEHVADVVRRCPHHQN----L-DAA
EYRSHLIRMEGSQRAQYFEDAYTKRQSVTVPYEPPQLGSEMTTILLSFMCNSSCMGGMNRRPILTI
MTLETPDGLILGRRCFEVRICACPGRDRKTEEENNTKKQNGTKETKK----------RKSTPVAAAA-P
VKKPKP----------TSSTDGDDKEIIYLAVRGRERYEILKKLNDGLELLEKEGKNK
SNVA----------VKQDVPLPSSGKRLLLKEERSDTD----------

>SwanGoose_wild_tp53

----MDPVPDLPESQGSFQELWETVY----PPLETLSLPTVNEPTGSWVATGDMFLL----------DQDL------
----------SGTFDDKIFD--------IPIEPVPTNEVNPPPTTVPVTTDYPGSYELELRFQKSGTAKSVTS
TYSETLNKLYCQLAKTSPIEVRVSKEPPKGAILRATAVYKKTEHVADVVRRCPHHQN----E-DSVEH

RSHLIRVEGSQLAQYFEDPYTKRQSVTVPYEPPQPGSEMTTILLSYMCNSSCMGGMNRRPILTILT
LET-EGLVLGRRCFEVRICACPGRDRKTEEESRQKTQPK-----K----------RKVTPNTSSS-K-RKKSH--
----------SSGEEEDNREVFHFEVYGRERYEFLKKINDGLELLEKESKSKNKD---------
---------SGMVPSSGKKLKSN----------

>RattleSnake_wild_tp53

----------MD
QAVEEDPDPPLSPPLSQETFKQMWCKVVHPFWEL------NADSTQ----QTTIPYLVEGGSSTDQTNFG
LSAPPVSDGSPLGALAEGYPGGEGYPTLALSDFLEN-----S-QFVPLAPQDYGAGESCPIPPTEDYPG
EFGFNLEFEQSGTAKSVTYTFSPQLNKLFCQMSKTCKVLIRMSGAPPPGSVVRAMAVYKRSEHM
SEVVRRCPHHERHPEHN-DGMIPPDHLIRVEGNPLAHYHTDPGTKRHSVWVLYQKPQVGMACTI
VLYNYMCNSSCMGGMNRRPILTVLTLETEQGEVLGRRCFEVRVCACPGRDRRSEELSFQQKGKS
LENTKKAP----------LQGPHPCASSGRPVQVA----------PE----------EEAPECESGGTSEPKVYTL
VIRNPRHYAILKELLEGLECKEACQKNGEGCKSGK---SVLKSRK--RAKLTSPPPKKFRVKDESQDS
N----------

>KomodoDragon_wild_tp53

----------M
DQESELESSLSQIMSQETFQDLWNTVLQPGSST------F-PDPCQI--ED------V----------TLGLPP------SG
ELVD-----LSFPEEGR------YSRAEEDGNPIPPLETVGPIQTSSVPSTEDYVNEYGFELVFEQSGVSKSI
TCTYSPILKKLFCQLAKTCKVLIKVDLPPPAGSFVRAMAVYKKSEHIADVVKRCPHHERATEYN-D
GVAPADHLIRVEGNRQARYISSEDTQRHSVIVPYEMPQVGSAYSTVLYNFMCNSSCMGGMNRRAI
LTIITLETPQGQLLGRRCFEVRVCACPGRDRRTEEENYQKAPGRD----------KKLKKALP-QGSVSE
NSKKVAKT----------GTNASESGTYTLQIRNRKHYQILKRLLEALEFQELWQPT
EAESQK--PAKGLLKAQKQDASPSSFRMEHCSSAFQEESDEDEEPGPVQKKLKKENED

>Axolotl_wild_tp53

----------MV-
----ESEAAMDSLPMSQETFSQLWGSLPETLLQ-------STDES---FWHQDFDYALD----------MGPQPQ--
----L---QQ-----ED----VSSTLNNDVLGAYEVPLPEAPPL-ESPAATSTVPSTEDYPGTYNFKLTFQQSG
TAKSVTSTYSPDLNKLFCQLAKTCPVQMTVSNPPPPGAIVRATAVYKKSQHIAEVVKRCPHHERS
GDPN-DGHAPPSHLIRVEGNFQAEYMENVTTRRHSVVVPYETPQVGSGCTTVLYNYMCNSSCMG
GMNRRPILTIITLESNDGQLLGRRCFEVRVCACPGRDRKTEEENFHNKKKPSKNS---VNSGTKRTI
K-DTTLTEADAEHP----K----------KKATEEEEIYTLQIRGRERYEMLKKLNDAL
ELKDLIPQSDQEKIKQKFQKPR-----KERENLAPKNGKKLLVKDETQDSD----------

>ChinookSalmon_wild_tp53

----------MA
D------LAENVSLPLSQESFEDLWKMNLNLMEVQ------PPVTE---SWVG-YDNFMM----------EAPLQ--
----------VE----FDPSLFEVSATE-PAPQPSISTLDTGSPPTSTVPTTSDYPGALGFQLRFLQSSTAK
SVTCTYSPDLNKLFCQLAKTCPVQIVVDHPPPPGAVVRALAIYKKLSDVADVVRRCPHHQSTSEN
N-EGPAPRGHLVRVEGNQRSEYMEDGNTLRQSVLVPYEPPQVGSECTTVLYNFMCNSSCMGGMN
RRPILTIITLETQEGQLLGRRSFEVRVCACPGRDRKTEEINLKKQQETILETKTKPAQGIKR-AMKEA

SLPAARPEASKKTKS----------SPAVSDDEIYTLQIRGKEKYEMLKKFNDSLELS
ELVPVADADKYRQKRLTKRVAK-RDF--GVGLKKGKKLLVKEEKSDSD----------

>ChannelCatfish_wild_tp53

----------ME
GN---GERDTMMVEPPDSQEFAELWLRNLIVRDNS------L-WGK---EEEI----PDD----------LQEVP-----
----------CDVLLSDMLQPQS------SSSPPTSTVPVTSDYPGLLNFTLHFQESSGTKSVTCTY
SPDLNKLFCQLAKTCPVLMAVSSSPPPGSVLRATAVYKRSEHVAEVVRRCPHHERSNDSS-DGPAP
PGHLLRVEGNSRAVYQEDGNTQAHSVVVPYEPPQVGSQSTTVLYNYMCNSSCMGGMNRRPILTII
TLETQDGHLLGRRTFEVRVCACPGRDRKTEESNFKKQQEPKTSGKTLT--KR--SMK----DPPSHPEA
SKKSK----------N-SSSDDEIYTLQVRGKERYEFLKKINDGLELSDVVPPADQE
KYRQKLLSKTCRK-ERDGAAGEPKRGKKRLVKEEKCDSD----------

>ZebraFish_wild_tp53

---------MAQNDSQEFAELWEKNLISIQPP------G-GGS---CWDI----IND----------EEYLP----------
GS----FDPNFFENVL--EEQPQP------STLPPTSTVPETSDYPGDHGFRLRFPQSGTAKSVTCTYSPDL
NKLFCQLAKTCPVQMVVDVAPPQGSVVRATAIYKKSEHVAEVVRRCPHHERTPD-G-DNLAPAGH
LIRVEGNQRANYREDNITLRHSVFVPYEAPQLGAEWTTVLLNYMCNSSCMGGMNRRPILTIITLE
TQEGQLLGRRSFEVRVCACPGRDRKTEESNFKKDQETKTMAKTTT--GTKRSLVKESSSATLRPEG
SKKAKG----------SSSDEEIFTLQVRGRERYEILKKLNDSLELSDVVPASDAE
KYRQKFMTKN-KK-EN-RESSEPKQGKKLMVKDEGRSDSD----------

>GoldFish_wild_tp53

----------MAESQEFADLWEKNLIST--P------E-AGT---CWEL----IND----------EYLP----------SP--
--FDPNIFDNVQ--TEQPQP------STSPPTASVPVATDYPGEHGFKLGFPQSGTAKSVTCTYSSDLNKL
FCQLAKTCPVQMMVDVAPPQGSVVRATAIYKKSEHVAEVVRRCPHHERTPD-G-DGLAPAAHLIR
VEGNSRALYREDEVNLRHSVVVPYEAPQLGAEFTTVLFNFMCNSSCMGGMNRRPILTIITLETHD
GQLLGRGSFEVRVCACPGRDRKTEESNFRKDQETKTSGKTPS--SNKRSLTKESTSSVPRPEGSKK
AKL----------S-GSSDEEMYNLQVRGKERYEILKMINDSLELSDVVPPSEIDRYR
QKILAKG-KK-EKDGQTPEPKRGKKLLVKDEKSDSD----------

>CommonCarp_wild_tp53

----------MAESQEFADLWERNLIST--Q------E-AGT---CWEL----IND----------EQYLP----------SS-
---FDPNIFDNVL--TEQPQP------STSPPTASVPVATDYPGEHGFKLGFPQSGTAKSVTCTYSSDLNKL
FCQLAKTCPVQMVVDVAPPQGSVIRATAIYKKSEHVAEVVRRCPHHERTPD-G-DGLAPAAHLIRV
EGNSLALYREDEVNSRHSVVVPYEAPQLGAEFTTVLFNFMCNSSCMGGMNRRPILTIITLETHDG
QLLGRGSFEVRVCACPGRDRKTEESNFRKDQETKTLGKTPS--TNKRSLTKESTSSVPRPEGSKKA
KL----------S-GSSDEEIYTLQVRGKERYEMLKKINDSLDLSDVVPPSEMDRYR
QKLLTKG-KK-EKDGQTPEPKRGKKLMVKDEKSDSD----------

>GreatWhiteShark_wild_type

---MSESQLDEPLSQETFRELWNQLEVPSANV------GLENE-LQIWDNEFSGLEL----------AMEELD---
---N---NP-----LEFPVLPDNPLPYP--SSSQAG----PATDIHVAAPCTVLATTEYPGPHEFQLQFQQSSTA
KSVTCTYSPSLNKLFCQLAKTCPVQVVVASVPPTGTLLRATAVYKKPEHVAEVVKRCPHHERG-S
ET-DGPAPPSHLIRVEANSRARYAEDEHTKRQSVIVPYESPQVGSDYTTVLYNFMCNSSCMGGMN
RRPILSILTLETPDGHLLGRRCFEVRVCACPGRDRKSEEENLKRQQENSMVKSGGS--ATKRTIKEV
SQ-ATTSPDSRKKKA----------LSDDEVFTLQVRGRERYELMKKLNEALEIS
ELIPTGVIEAYKQQQKHRLKASHKKEKESTEIKNGKKLLVKDERDSD----------

>GiantTortoise_wild_tp53

MIWFDLYNPGHKPALKSSLQAASSPSLHYAKSLALHEAAPLCAAPWGARQHPLPLCPPPGQPDD
MEEQDGDAGLMGPAVTWPRHPRNASWLPGNPQLVHVDAKCPGETGVQISAERTAGMEPMLDPG
LEPMLDPGLEPPLSQESFSDLWSMMARRTAQD------SNTQDCPGLYSLPDPDLSL----------DLGLSD
------S----------ADPSLLLPQAGGSDGGWGLPDPAPEPPPTSATVPSTEDYAGEHGFELAFQQSG
TAKSVTCTYSPQLNKLYCQLAKTCPVQIRTASQPPAGSIIRATAVYKKSEHVAEVVRRCPHHERCE
EYR-DGVAPARHLIRIEGNQQAHYYDDENTKRQSVTVPYETPQLGSDCTTVLYNFMCNSSCMGG
MNRRPILAIITLEGRHRQLLGRRCFEVRVCACPGRDRRTEEENFCKKLAGRVLNGAGA--HKGGG
AKRALQATMETAENPKKLVVS----------SE----------KEVFLLEVHGRKRYMMLKEIN
DALEMAAAKQLGEPESHRNATPSRLLKTRKESADGVLPRSGKKLLVKEEDSE----------

>Chinese-Alligator_wild_tp53

MESIMDPDLDPPLS-QPFLDFWNVLDNNVRSI------PKEQA--ELWDPQDL---V----------LG-LPD------
L----------GDLPLLEELEGAPVAGLGREAPPPGALPTSSIVPSTEDYPGAHGFEVAFQPSGTAKS
VTCTYSPVLNKLFCQLAQSCPVQVRVAQAPPPGAMIRAGAVYKKAEHVAEVVRRCPHHERSAEH
S-DGVAPAQHLIRVEGNPQAQYCHDETTKRHSVTVPYTPPEVGSDSTTVLYNFMCNSSCMGGMN
RRPILAILTLETKSGQLLGRRCFEVRICACPGRDRKTEEENLRNK---------AA--TTGGGAKRALKVP
TDDLPNPKKRVPN----------PS----------TEIFTLQIRGHERYEMFKKLNEGLEALDGQE
ARAE-DPGIRSPKPLLKARR--AKGLALVSCKKLLVKDESQDSD----------

>African-Ostrich_wild_tp53

MAEELEPLLEPPLT-EGFLDLWNMLPDNMHSL------PLPDE-PAAWDLAPL---A----------LPP----------
----------EA--APEVVPPQDPGVRAPPLVPSTEDYGGRYDFRLGFLQSGTAKSVTCTYSP
ELNKLYCQLAKPCPVQVRVGARPPPGARLRASAVYKKSEHVAEVVRRCPHHERCGPPG-DGLAP
AQHLIRVEGNPQARYHDDETTKRHSVSVPYEPPEVGSDCTTVLYNFMCNSSCMGGMNRRPILAIL
TLEGPGGQLLGRRCFEVRVCACPGRDRKTEEENFRKK---------GG--A-KGGAKRGEGGPRPLAAPS
RPHLLA----------RKPRLS-----RRPRPLLTPSPAPSPAPARTGFFFFFF----------
----

>Goshawk_wild_tp53

----------MG---A----------EPPGSP------C----------GPPPGG-----VP-
-VPPLPADPPPPMPPSPVVPSTEDYGGHHNFRLGFLEAGTAKSVTCTYSPELNKLYCRLAKPCPVQ

VRVGVPPPPGALLRAVAVYKKSEHVAEVVRRCPHHERCGGPG-DGLAPAQHLIRVEGNPQARYHD
DETTKRHSVAVPYEPPEVGSDCTTVLYNFMCNSSCMGGMNRRPILAILTLEGPGGQILGRRCFEVR
VCACPGRDRKIEEENYRKR---------GG--A-KGGAKRALSPAAEAPESSKKRVLN----------PDP-------
----------EVFCLQVRGRRRYEMLKEINEALEAAEGGGGTAT-AGEGSPPA------G--GRGWCR
AAGRSCC-------------------------

>CommonMallard_wild_tp53

MAEELEPLLEP--P-EIFLELWNMLPDNMHSL------SPPDD-PLAVQDLCP---L----------EPSEPP------P--
----------GPPP------STE--PPPAAPPEPPRASPSSMVPSTEDYGGHYDFQLGFQETGTAKSVTCTYS
PVLNKLYCRLAKPCPVQVRVGAAPPPGAVLRAVAVYKKSEHVAEVVRRCPHHERNGEGT-DGLAP
AQHLIRVEGNPQARYHDDETTKRHSVAVPYEPPEVGSDCTTVLYSFMCNSSCMGGMNRRPILAIIT
LEGPGGQLLGRRCFEVRVCACPGRDRKIEEENFRKR---------GG--A-GGGAKRGARPPPLRDAEGD
QRRPA----------DG----------RGGGGATA-V-EGAPPA------G--GG
AAAP-LREEAAAQGGAPGLRLTTPPVLATPPSPA--

>Chicken_wild_tp53

MAEEMEPLLEP--T-EVFMDLWSMLPYSMQQL------PLPED-HSNWQELSP---L----------EPSDPP------
P----------PPPPPPLPLAAAA--PPPLNPPTPPRAAPSPVVPSTEDYGGDFDFRVGFVEAGTAKSV
TCTYSPVLNKVYCRLAKPCPVQVRVGVAPPPGSSLRAVAVYKKSEHVAEVVRRCPHHERCGGGT-
DGLAPAQHLIRVEGNPQARYHDDETTKRHSVVVPYEPPEVGSDCTTVLYNFMCNSSCMGGMNR
RPILTILTLEGPGGQLLGRRCFEVRVCACPGRDRKIEEENFRKR---------GG--A-GGVAKRAMSPPTE
APEPPKKRVLN----------PDN----------EIFYLQVRGRRRYEMLKEINEALQLAEGGSA
PRP-S-KGRRVK------V--EGPQPS-CGKKLLQKGSD----------

>Japanese-Medaka_wild_tp53

MAEELEPLLEPPLT-EVFLDLWNMLPDNMHSL------SPPDD-PLAVQDLCP---L----------EPSEPP------
P----------GAPP------GAE--PPPAAPPEPPRASPSPMVPSTEDYGGHYDFQLGFLEAGTAKSVTC
TYSPVLNKLYCRLAKPCPVQVRVGAAPPPGAVLRAVAVYKKSEHVAEVVRRCPHHERNGEGA-D
GLAPAQHLIRVEGNPQARYHDDETTKRHSVTVPYEPPEVGSDCTTVLYSFMCNSSCMGGMNRRPI
LAILTLEGPGGQLLGRRCFEVRVCACPGRDRKIEEENFRKR---------GG--A-GGGAKRALSPPAKAP
ETPKKRVLN----------PDN----------EIFCLQVHGRRRYEMLKEINDALQMAEGGAAP
RP-S-KGHRPR------G--EGPLPR-SGKKLLLKGEPQDSD----------

Source code

```
---
title: "Phylogenetic_Tree"
author: "Ryder Sabale"
date: "`r Sys.Date()`"
output:
  pdf_document: default
  word_document: default
```

```{r setup, include=FALSE}
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
```

```{r}
# Loading necessary packages
library(seqinr)
library(ape)
library(msa)
```

```{r}
# Function uses basic workflow between packages from:
# Bryan Temogoh of Applied EPI
FASTA_to_Phylo_Tree <- function(file, sequence_type = c("DNA",
"RNA", "AA", "Other"), name = file) {

  # Read the fasta files using msa package
  if(sequence_type == "DNA") {
    # If the sequence is in DNA nucleotides (A,C,G,T)
    sequences <- readDNAStringSet(file)
  } else if (sequence_type == "RNA") {
    # If the sequence is in RNA nucleotides (A,C,G,U)
    sequences <- readRNAStringSet(file)
  } else if (sequence_type == "AA") {
    # If the sequence is in amino acids
    sequences <- readAAStringSet(file)
  } else {
    # The reader will just read each characters for what it is,
regardless of what means
    sequences <- readBStringSet(file)
  }

  # Align and print the sequences using the msa package,
  # type arguement not need to be set since if statement above
ensures seqeunces will be an XStringSet Class Object, which auto
fills type for this function

  msa <- msa(sequences, method = "ClustalOmega")
  print(msa)

  # Turn the unique MSA Class objects from the msa package into
list vectors useable by the seqinr package
  alignment <- msaConvert(msa, type = "seqinr::alignment")
```

```
  # Compute a distance matrix from the alignment using the
sequir package
  distance_matrix <- dist.alignment(alignment, matrix =
"identity")

  # Use an applied version of neighbor joining from the ape
package to construct a tree from the distance matrix
  tree <- bionj(distance_matrix)

  # Plot the tree using base R
  plot(tree, main = name)
}
```

```{r}
mammal_sequences <- "C:/Users/ryder/CS-123A/mammalAA.fasta"
nonmammal_sequences <-
"C:/Users/ryder/CS-123A/nonMammalAA.fasta"

FASTA_to_Phylo_Tree(mammal_sequences, sequence_type = "AA", name
= "Mammal Phylogenetic Tree")
FASTA_to_Phylo_Tree(nonmammal_sequences, sequence_type = "AA",
name = "Non-Mammal Phylogenetic Tree")
```

---
title: "TranslationintoAA.py"
author: "Martin Sanchez"

```
# standard codon table for translation; used as a python dictionary

codonTable = {

     'ATA': 'I', 'ATC': 'I', 'ATT': 'I', 'ATG': 'M',

     'ACA': 'T', 'ACC': 'T', 'ACG': 'T', 'ACT': 'T',

     'AAC': 'N', 'AAT': 'N', 'AAA': 'K', 'AAG': 'K',

     'AGC': 'S', 'AGT': 'S', 'AGA': 'R', 'AGG': 'R',

     'CTA': 'L', 'CTC': 'L', 'CTG': 'L', 'CTT': 'L',

     'CCA': 'P', 'CCC': 'P', 'CCG': 'P', 'CCT': 'P',

     'CAC': 'H', 'CAT': 'H', 'CAA': 'Q', 'CAG': 'Q',

     'CGA': 'R', 'CGC': 'R', 'CGG': 'R', 'CGT': 'R',
```

```
    'GTA': 'V', 'GTC': 'V', 'GTG': 'V', 'GTT': 'V',
    'GCA': 'A', 'GCC': 'A', 'GCG': 'A', 'GCT': 'A',
    'GAC': 'D', 'GAT': 'D', 'GAA': 'E', 'GAG': 'E',
    'GGA': 'G', 'GGC': 'G', 'GGG': 'G', 'GGT': 'G',
    'TCA': 'S', 'TCC': 'S', 'TCG': 'S', 'TCT': 'S',
    'TTC': 'F', 'TTT': 'F', 'TTA': 'L', 'TTG': 'L',
    'TAC': 'Y', 'TAT': 'Y', 'TAA': '*', 'TAG': '*',
    'TGC': 'C', 'TGT': 'C', 'TGA': '*', 'TGG': 'W',
}
def translate(ntSeq):
    """
    this fucntion translates a nucleotide sequence into an amino
acid sequence using the nromal codon table.
    nucleotide_seq (arg)(str): The nucleotide sequence.
    returns str: The translated amino acid sequence without stop
codons.
    """
    proteins = []
    # translate current codon
    for i in range(0, len(ntSeq) - len(ntSeq) % 3, 3):
    current = ntSeq[i:i + 3]
    aa = codonTable.get(current, 'X')  # 'X' for unknown codons
    if aa == '*':  # skip the stop codons
        break
    proteins.append(aa)# append translated amino acid
    return ''.join(proteins)


def process_fasta(inputf, outputf):
    """
```

```
     processes a .fasta file and translates nucleotide sequences
into amino acid sequences.

     input_fasta (args)(str): Path to the input FASTA file.

     output_fasta (args)(str): Path to save the output FASTA file.

     """

     with open(inputf, 'r') as infile, open(outputf, 'w') as
outfile:

     currentID = None

     currentSEQ = []

     for line in infile:

          line = line.strip()

          if line.startswith(">"):  # FASTALINE

               # if we have a sequence collected, process and write
it

               if currentID is not None:

                    translated_seq = translate(''.join(currentSEQ))


outfile.write(f"{currentID}\n{translated_seq}\n")

               #new seq

               currentID = line

               currentSEQ = []

          else:

               currentSEQ.append(line.upper())

     #last seq

     if currentID is not None:

          translated_seq = translate(''.join(currentSEQ))

          outfile.write(f"{currentID}\n{translated_seq}\n")


inputFILE =
"/Users/martiin/VSCode/2024JavaPersonal/MS2024/src/nonMammal_tp53_nt.
fasta"
```

```
outputFILE =
"/Users/martiin/VSCode/2024JavaPersonal/MS2024/src/output.fasta"

process_fasta(inputFILE, outputFILE)

print(f"Translation finished, saved to {outputFILE}")


_
title: "MSA_CS123.py"
author: "Martin Sanchez"



from Bio.Seq import Seq

from Bio.SeqRecord import SeqRecord

from Bio import SeqIO

# Scoring function:

match = 1

mismatch = -1

gap = -2


def needlemanWunsch(seq1, seq2):

     """

     fucntion will perform alignment using the NeedlemanWunsch
algorithm.

     seq1, seq2(both arguments): The two current sequences to align.

     return value::

     Two aligned sequences with gaps introduced where necessary.

     REFERENCES:
https://stackoverflow.com/questions/63120727/needleman-wunsch-algorit
hm-for-two-sequences-of-different-length

     """

     s1Length = len(seq1)# initialize lengths of sequences 1 and 2.

     s2Length = len(seq2)

     # STEP 1: DRAW GRID (relative to sequence lengths)
```

```
    Matrix2D = [[0] * (s2Length + 1) for _ in range(s1Length + 1)]

    # STEP 2: INITIALIZE THE GRID

    # initializing the first rows anf columns with necessary gap
penalties.

    for i in range(1, s1Length +1):

    Matrix2D[i][0] = i*gap

    for j in range(1, s2Length +1):

    Matrix2D[0][j] = j*gap

    #STEP 3: FILL IN THE MATRIX.

    for i in range(1, s1Length +1):

    for j in range(1, s2Length + 1):

         deletion = Matrix2D[i -1][j] +gap  # Deletion (gap in
seq2)

         insertion = Matrix2D[i][j -1] +gap  # Insertion (gap in
seq1)

         matchScore = Matrix2D[i -1][j -1] + (match if seq1[i-1] ==
seq2[j-1] else mismatch)

         Matrix2D[i][j] = max(deletion, matchScore,insertion)  #
Pick the best score


    # STEP 4: TRACEBACK

    align1 = []

    align2 = []

    i = s1Length

    j = s2Length


    while j and i > 0:

    currScore = Matrix2D[i][j]

    # statement handles match and mismatch case

    if Matrix2D[i - 1][j - 1] + (match if seq1[i - 1] == seq2[j -
1] else mismatch)==currScore:
```

```
            align1.append(seq1[i - 1])
            align2.append(seq2[j - 1])
            j=j-1
            i=i-1
      # gap in seq2
      elif  Matrix2D[i - 1][j] + gap==currScore:
            align2.append("-")
            align1.append(seq1[i - 1])
            i=i-1
      # gap in seq1
      else:
            align2.append(seq2[j-1])
            align1.append("-")

            j=j-1
      # add all remaining gaps if ran out of characters in one
sequence
      while i > 0:
      align2.append("-")
      align1.append(seq1[i - 1])
      i = i-1
      while j > 0:
      align1.append("-")
      align2.append(seq2[j - 1])
      j=j-1
      # reverse the sequences to return in the correct order
      align1 = ''.join(reversed(align1))
      align2 = ''.join(reversed(align2))
      return align1, align2
```

```
def commonMotifs(sequences):
    """
    identify the common motif sequence from list of aligned
sequences.
    sequences(argument): List of aligned sequences with gaps.
    return value:
    A single consensus sequence based on the most common character
at each position.
    """
    length = len(sequences[0])
    motifs = []
    #loop to extract the ith column across all seqs. find the
commonly occurring chars
    for i in range(length):
    column = [seq[i] for seq in sequences]
    mostCommon = max(set(column), key=column.count)
    motifs.append(mostCommon)
    return ''.join(motifs)


def SeqGapping(sequences):
    """
    function serves to verify all sequences are of equal length.
    sequences(arg): lists of aligned sequences with varying
lengths.
    return a List of sequences padded with gaps to match the
longest sequence.
    """
    maxLength = max(len(seq) for seq in sequences)  #determine the
max length of all seqs
```

```
    gappedSeqs = [seq.ljust(maxLength, "-") for seq in sequences]
# if seqs are less than max, then add gaps
    return gappedSeqs


def alignTheTails(sequences):
    """
    function to align the trailing egions of all sequences based on
the longest motif.
    sequences(arg): List of aligned sequences with varying end gaps
    returns a list of sequences with the ends aligned to the
longest motif.
    """
    longTail = max([seq.rstrip('-') for seq in sequences], key=len)
    alignTails = []
    for seq in sequences:
    stripped_seq = seq.rstrip('-')  #remove trailing gaps
    aligned_seq = stripped_seq.ljust(len(longTail), '-')
    alignTails.append(aligned_seq)
    return SeqGapping(alignTails)  # add more gaps just in case.


def alignAlongMotifs(consensus, sequences,):
    """
    fucntion to align all sequences to the given common motif
    consensus(arg): The consensus sequence to align against.
    sequences(arg): List of sequences to align
    returns a list of sequences aligned to the consensus.
    """
    aligned_sequences = []
    for seq in sequences:
    _, aLIGNEDSEQ = needlemanWunsch(consensus, seq)
```

```
    aligned_sequences.append(aLIGNEDSEQ)
    return SeqGapping(aligned_sequences)


def MSA(sequences, iterations=30):
    """
    does multiple sequence alignmentiteratively.
    sequences(arg): List of sequences to align.
    iterations(arg): Number of refinement iterations to perform.
    RETURNS Final list of aligned sequences.
    """
    alignedSEQUENCES = [sequences[0]]  # begin with the first
sequence
    for seq in sequences[1:]:
    for alignedSeq in alignedSEQUENCES:
         alignedSeq, seq = needlemanWunsch(alignedSeq, seq)  #
align each sequence progressively
    alignedSEQUENCES.append(seq)
    #refine alignment
    for _ in range(iterations):
    alignedSEQUENCES = SeqGapping(alignedSEQUENCES)  #add padding
if necessary
    consensus = commonMotifs(alignedSEQUENCES)
    alignedSEQUENCES = alignAlongMotifs(consensus, sequences)  #
realign
    alignedSEQUENCES = alignTheTails(alignedSEQUENCES)
    return alignedSEQUENCES


def format(alignedSEQs, blkSize=100):
    """
```

```
     fucntion to format the final alignment in a Clustal-like style
for readability.
     alignedSEQa(args):list of aligned sequences.
     blksize(args): num of characters to display per block.

     prints the formatted alignment with a consensus line.
     """
     seqlen = len(alignedSEQs[0])  # length of the aligned sequences
     motif = commonMotifs(alignedSEQs)
     # print
     for start in range(0, seqlen, blkSize):
     end = start + blkSize
     print()  #new line
     for idx, seq in enumerate(alignedSEQs):
          print(f"Seq{idx + 1:<3} {seq[start:end]}")
     print(f"   {motif[start:end]}")
# load sequences from local .FASTA file
fastaFile =
"/Users/martiin/VSCode/2024JavaPersonal/MS2024/src/mammalAA.fasta"
sequences = [str(record.seq) for record in SeqIO.parse(fastaFile,
"fasta")]
#execute MSA with iterative refinement and tail alignment
finalAlignment = MSA(sequences)
# print the alignment in Clustal-like format
format(finalAlignment)
# save the aligned sequences to a new FASTA file
with open("alignedSequences-CS123.fasta", "w") as output:
     for idx, aligned_seq in enumerate(finalAlignment):
     record = SeqRecord(Seq(aligned_seq), id=f"Seq{idx+1}")
     SeqIO.write(record, output, "fasta")
```