Bioinformatics Mini Project 2024

Part 1 – The ORF Detector

An Introduction to ORFs:

An open reading frame (ORF) is defined as a start codon followed by a downstream in-frame stop codon. The sequence of these will typically vary depending on whether the DNA itself is spliced or not (i.e. whether it contains regions of non-coding DNA). While ORFs vary in length, standard ORFs tend to be defined as being composed of at least 100 codons (Claverie et.al, 1997) and small ORFs (smORFs) are defined as being less than 100 codons in length (Kute et.al, 2022).

The ORF Detector:

Python code:

```
sequence = SeqIO.read("part1_sequence.fasta", "fasta")

# Printing the sequence and its ID
print("Sequence:", sequence.seq)
print("ID:", sequence.id)

def find_all_ORFs(sequence):
    ORFs = []
    #finding reverse complement and iterating over 3 frames for both sequences
    for strand, nuc in [(+1, sequence.seq), (-1, sequence.seq.reverse_complement())]:
        for frame in range(3):
            length = 3 * ((len(sequence)-frame) // 3) # Multiple of three for codons
            for i in range(frame, length, 3):
                codon = nuc[i:i+3]
                if codon in ['ATG', 'AUG']:
                    for j in range(i+3, length, 3):
                        if nuc[j:j+3] in ['TAA', 'TAG', 'TGA', 'UAA', 'UAG', 'UGA']:
                            ORFs.append(nuc[i:j+3])
                            break
    ORFs.sort(key=len, reverse=True)
    return ORFs  # Now returning a list of all ORFs, in order of size

ORFs = find_all_ORFs(sequence)

# Create a table to input results into
table = PrettyTable()

# Adding columns to the table
table.field_names = ["ORF Length", "ORF Sequence", "Amino Acid Sequence"]

# saving ORFs as fasta files and adding the translated amino acid sequence for each ORF into the table
for i, orf in enumerate(ORFs, 1):
    amino_acid_sequence = orf.translate(to_stop=True)
    table.add_row([len(orf), orf, amino_acid_sequence])
    record = SeqRecord(orf, id=f"ORF{i}", description=f"ORF{i} from sequence {sequence.id}")
    SeqIO.write(record, f"ORF{i}.fasta", "fasta")

# Print results in the table
print(table)
```

Imported Packages:

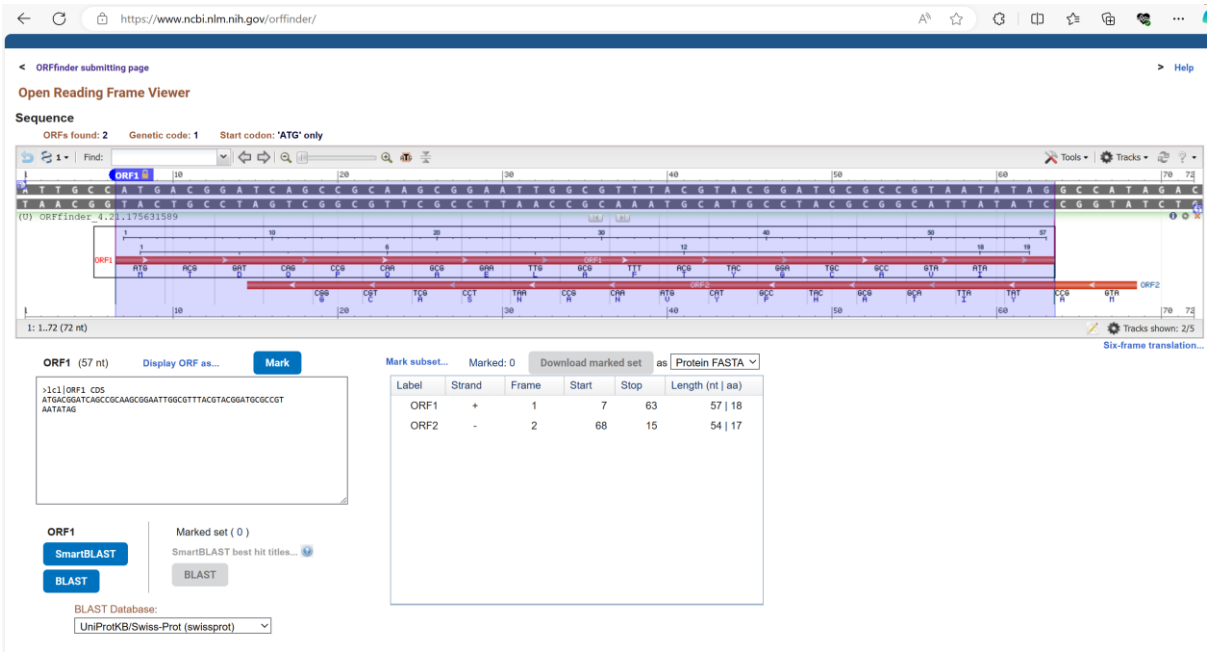
```
#importing libraries
from Bio.Seq import Seq
from Bio import SeqIO
from Bio.SeqRecord import SeqRecord
from prettytable import PrettyTable
```

Testing with the trial sequence:

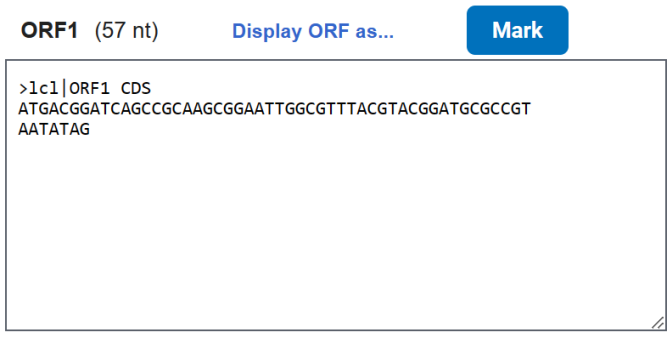
Trial sequence (part1_sequence): ATTGCCATGACGGATCAGCCGCAAGCGGAATTGGCGTTTACGTACGGATGCGCCGTAATATAGGCCATAGAC

Expected outputs:

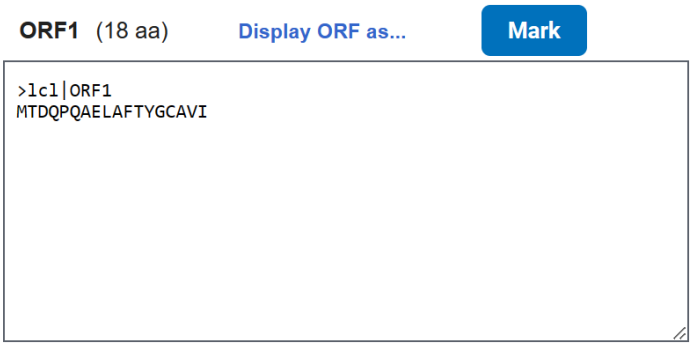
The online ORF finder from the National Center for Biotechnology Information website (ORFfinder Viewer - NCBI (nih.gov)) was used to check the output of the ORF finder using the 'part1_sequence' fasta file as a trial sequence.



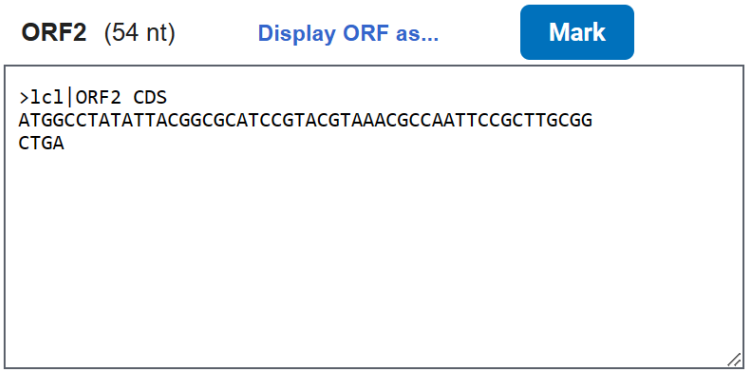
Expected ORF 1:



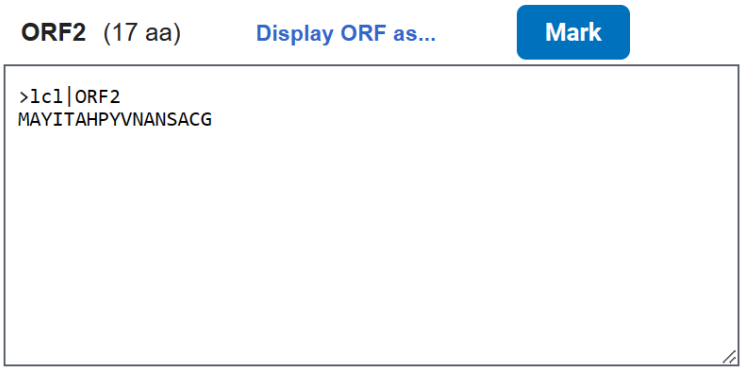
Expected Amino acid sequence for ORF 1;



Expected ORF 2:



Expected Amino acid sequence for ORF 2:



Actual output:

Actual ORF and Amino Acid output:

```
Sequence: ATTGCCATGACGGATCAGCCGCAAGCGGAATTGGCGTTTACGTACGGATGCGCCGTAATATAGGCCATAGAC
ID: part1_sequence
+------------+-----------------------------------------------------------+--------------------+
| ORF Length |                        ORF Sequence                       | Amino Acid Sequence |
+------------+-----------------------------------------------------------+--------------------+
|     57     | ATGACGGATCAGCCGCAAGCGGAATTGGCGTTTACGTACGGATGCGCCGTAATATAG |  MTDQPQAELAFTYGCAVI |
|     54     |   ATGGCCTATATTACGGCGCATCCGTACGTAAACGCCAATTCCGCTTGCGGCTGA  |  MAYITAHPYVNANSACG  |
+------------+-----------------------------------------------------------+--------------------+
```

The actual output for the ORF nucleotide sequence and Amino acid sequence fit the Expected outputs found by the NCBI ORF Finder tool. Therefore no edits needed to be made to the Python script after

this result. However, this ORF finder still had not been tested on larger sequences, so the BRCA2 cDNA sequence was input as a query (please see below).

Testing with the BRCA2 cDNA sequence:

Expected output:

It is expected that the longest ORF from this sequence will match the BRCA2 peptide sequence given in Ensemble ().

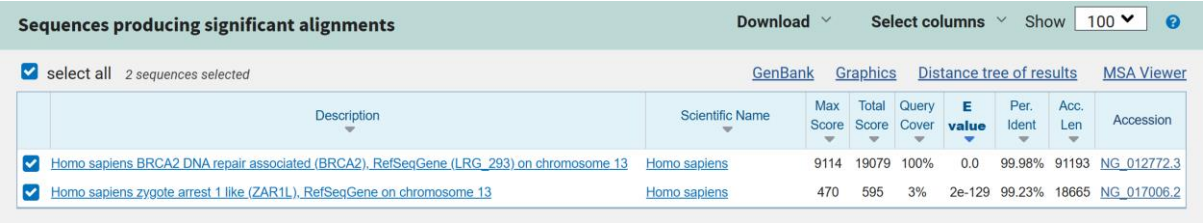
Actual output with the ORF finder:

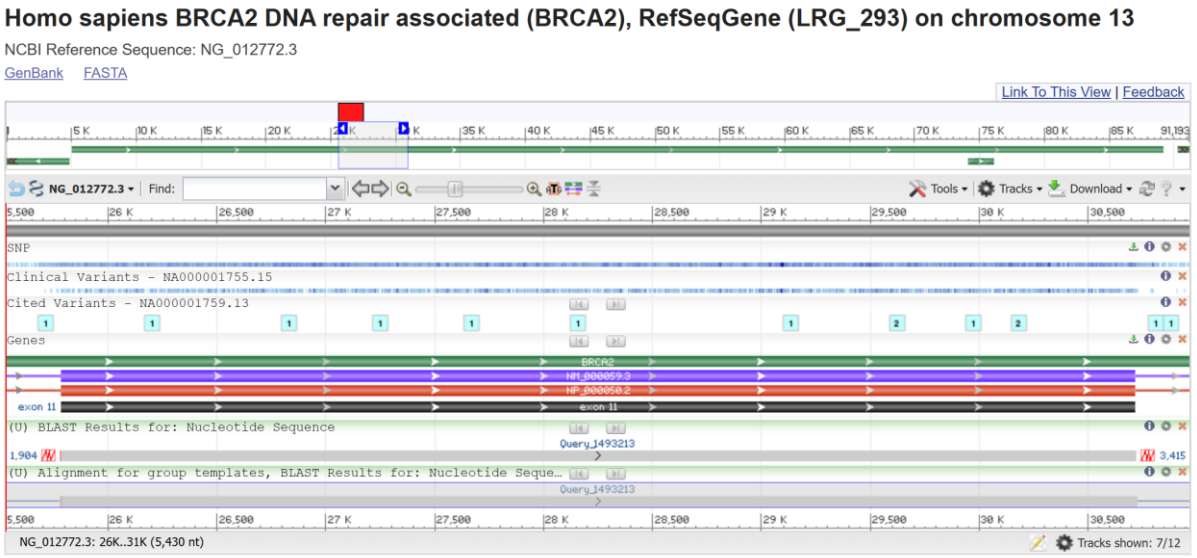
```python
#importing libraries
from Bio.Seq import Seq
from Bio import SeqIO
from Bio.SeqRecord import SeqRecord

#reading the fasta file
sequence = SeqIO.read("BRCA2_cDNA_ensemble.fa", "fasta")

def find_all_ORFs(sequence):
    ORFs = []
    #finding reverse complement and iterating over 3 frames for both sequences
    for strand, nuc in [(+1, sequence.seq), (-1, sequence.seq.reverse_complement())]:
        for frame in range(3):
            length = 3 * ((len(sequence)-frame) // 3) # Multiple of three for codons
            for i in range(frame, length, 3):
                codon = nuc[i:i+3]
                if codon in ['ATG', 'AUG']:
                    for j in range(i+3, length, 3):
                        if nuc[j:j+3] in ['TAA', 'TAG', 'TGA', 'UAA', 'UAG', 'UGA']:
                            ORFs.append(nuc[i:j+3])
                            break
    ORFs.sort(key=len, reverse=True)
    return ORFs  # Now returning a list of all ORFs, in order of size

ORFs = find_all_ORFs(sequence)

print()

# Printing the longest ORF and its length
if ORFs:
    print(f'Longest ORF: {ORFs[0]}')
    print()
    print(f'Length of longest ORF: {len(ORFs[0])}')

# Printing the total number of ORFs found
print()
print(f'Total ORFs found: {len(ORFs)}')
print()


# Saving the longest ORF as a fasta file: BRCA2_longest_orf.fasta
if ORFs:
    longest_orf = SeqRecord(ORFs[0], id="Longest_ORF", description="Longest ORF from BRCA2")
    SeqIO.write(longest_orf, "BRCA2_longest_orf.fasta", "fasta")
```

```
Length of longest ORF: 10257

Total ORFs found: 429
```

This code saves the longest ORF out of the 429 found as a fasta file (BRCA2_longest_orf.fasta) and also printing it as an output (for my own debugging/viewing benefit).

BLAST results for the BRCA2 gene longest ORF:





Comments:

As expected, the observed output matched the BRCA2 peptide sequence given in Ensemble. Therefore, as long as no unforseen bugs occur in the code, this script is able to detect the longest ORF and all ORFs within a sequence that is fairly long or reasonably short in length.

Possible extension and model limitations:

It cannot be expected that all ORFs found by the ORF finder will code for proteins. Whilst the output was expected for the human BRCA2 cDNA sequence, the result could have been different if the regular DNA sequence was used (pre-splicing). This current model of the ORF Detector cannot distinguish between introns and exons. Whilst it could be theoretically possible to implement some kind of command to bias the ORF finder against regions expressing typical intron sequence motifs (such as increased C/G repetition), (Xie et.al, 2023). these alterations would most likely have to be species specific due to the highly variable abundance of introns between species and would most likely be prone to bias (Zlotorynski et.al, 2019).

Part 2 – Decoding Dinosaur DNA

Introduction – Sequence 1

The goal of this section (Part 2, section 1) is to determine whether the Dinosaur DNA sequence from Michael Crichton's Jurassic Park is based on fact or fiction. To achieve this, the ORF finder coded for in Part 1 will be utilised as well as the NCBI BLAST+ online webservice.

Sequence-1: Finding 'Dinosaur' ORFs

Dinosaur DNA Sequence 1:
GCGTTGCTGGCGTTTTTCCATAGGCTCCGCCCCCCTGACGAGCATCACAAAAATCGACGCGGTGG
CGAAACCCGACAGGACTATAAAGATACCAGGCGTTTCCCCCTGGAAGCTCCCTCGTGTTCCGACC
CTGCCGCTTACCGGATACCTGTCCGCCTTTCTCCCTTCGGGAAGCGTGGCTGCTCACGCTGTACCT
ATCTCAGTTCGGTGTAGGTCGTTCGCTCCAAGCTGGGCTGTGTGCCGTTCAGCCCGACCGCTGCG
CCTTATCCGGTAACTATCGTCTTGAGTCCAACCCGGTAAAGTAGGACAGGTGCCGGCAGCGCTCT
GGGTCATTTTCGGCGAGGACCGCTTTCGCTGGAGATCGGCCTGTCGCTTGCGGTATTCGGAATCT
TGCACGCCCTCGCTCAAGCCTTCGTCACTCCAAACGTTTCGGCGAGAAGCAGGCCATTATCGCCG
GCATGGCGGCCGACGCGCTGGGCTGGCGTTCGCGACGCGAGGCTGGATGGCCTTCCCCATTATGA
TTCTTCTCGCTTCCGGCGGCCCGCGTTGCAGGCCATGCTGTCCAGGCAGGTAGATGACGACCATC
AGGGACAGCTTCAACGGCTCTTACCAGCCTAACTTCGATCACTGGACCGCTGATCGTCACGGCGA
TTTATGCCGCACATGGACGCGTTGCTGGCGTTTTTCCATAGGCTCCGCCCCCCTGACGAGCATCAA
AACAAGTCAGAGGTGGCGAAACCCGACAGGACTATAAAGATACCAGGCGTTTCCCCCTGGAAGC
GCTCTCCTGTTCCGACCCTGCCGCTTACCGGATACCTGTCCGCCTTTCTCCCTTCGGGCTTTCTCA
ATGCTCACGCTGTAGGTATCTCAGTTCGGTGTAGGTCGTTCGCTCCAAGCTGACGAACCCCCCGT
TCAGCCCGACCGCTGCGCCTTATCCGGTAACTATCGTCTTGAGTCCAACACGACTTAACGGGTTG
GCATGGATTGTAGGCGCCGCCCTATACCTTGTCTGCCTCCCCGCGGTGCATGGAGCCGGGCCACC
TCGACCTGAATGGAAGCCGGCGGCACCTCGCTAACGGCCAAGAATTGGAGCCAATCAATTCTTG
CGGAGAACTGTGAATGCGCAAACCAACCCTTGGCCATCGCGTCCGCCATCTCCAGCAGCCGCAC
GCGGCGCATCTCGGGCAGCGTTGGGTCCT

Results from the ORF finder:

ORF Length	ORF Sequence	Amino Acid Sequence
327	ATGACGACCATCAGGGACAGCTTCAACGGCTCTTA CCAGCCTAACTTCGATCACTGGACCGCTGATCGTCA CGGCGATTTATGCCGCACATGGACGCGTTGCTGGC GTTTTTCCATAGGCTCCGCCCCCCTGACGAGCATCA CAAACAAGTCAGAGGTGGCGAAACCCGACAGGAC TATAAAGATACCAGGCGTTTCCCCCTGGAAGCGCT CTCCTGTTCCGACCCTGCCGCTTACCGGATACCTGT CCGCCTTTCTCCCTTCGGGCTTTCTCAATGCTCACG CTGTAGGTATCTCAGTTCGGTGTAGGTCGTTCGCTC CAAGCTGA	MTTIRDSFNGSYQPNFDHWTADR HGDLCRTWTRCWRFSIGSAPLTSI TNKSEVAKPDRTIKIPGVSPWKRSP VPTLPLTGYLSAFLPSGFLNAHAV GISVRCRSFAPS
228	ATGCCGCACATGGACGCGTTGCTGGCGTTTTTCCAT AGGCTCCGCCCCCCTGACGAGCATCACAAACAAGT CAGAGGTGGCGAAACCCGACAGGACTATAAAGAT ACCAGGCGTTTCCCCCTGGAAGCGCTCTCCTGTTCC GACCCTGCCGCTTACCGGATACCTGTCCGCCTTTCT CCCTTCGGGCTTTCTCAATGCTCACGCTGTAGGTAT CTCAGTTCGGTGTAG	MPHMDALLAFFHRLRPPDEHHKQ VRGGETRQDYKDTRRFPLEALSCS DPAAYRIPVRLSPFGLSQCSRCRYL SSV
219	ATGGACGCGTTGCTGGCGTTTTTCCATAGGCTCCGC CCCCCTGACGAGCATCACAAACAAGTCAGAGGTGG CGAAACCCGACAGGACTATAAAGATACCAGGCGTT TCCCCCTGGAAGCGCTCTCCTGTTCCGACCCTGCCG CTTACCGGATACCTGTCCGCCTTTCTCCCTTCGGGC TTTCTCAATGCTCACGCTGTAGGTATCTCAGTTCGG TGTAG	MDALLAFFHRLRPPDEHHKQVRG GETRQDYKDTRRFPLEALSCSDPA AYRIPVRLSPFGLSQCSRCRYLSSV
213	ATGCGCCGCGTGCGGCTGCTGGAGATGGCGGACGC GATGGCCAAGGGTTGGTTTGCGCATTCACAGTTCTC CGCAAGAATTGATTGGCTCCAATTCTTGGCCGTTAG CGAGGTGCCGCCGGCTTCCATTCAGGTCGAGGTGG CCCGGCTCCATGCACCGCGGGGAGGCAGACAAGGT ATAGGGCGGCGCCTACAATCCATGCCAACCCGTTA A	MRRVRLLEMADAMAKGWFAHSQ FSARIDWLQFLAVSEVPPASIQVEV ARLHAPRGGRQGIGRRLQSMPTR
189	ATGGCGGACGCGATGGCCAAGGGTTGGTTTGCGCA TTCACAGTTCTCCGCAAGAATTGATTGGCTCCAATT CTTGGCCGTTAGCGAGGTGCCGCCGGCTTCCATTCA GGTCGAGGTGGCCCGGCTCCATGCACCGCGGGGAG GCAGACAAGGTATAGGGCGGCGCCTACAATCCATG CCAACCCGTTAA	MADAMAKGWFAHSQFSARIDWL QFLAVSEVPPASIQVEVARLHAPR GGRQGIGRRLQSMPTR
177	ATGGCCAAGGGTTGGTTTGCGCATTCACAGTTCTCC GCAAGAATTGATTGGCTCCAATTCTTGGCCGTTAGC GAGGTGCCGCCGGCTTCCATTCAGGTCGAGGTGGC CCGGCTCCATGCACCGCGGGGAGGCAGACAAGGTA TAGGGCGGCGCCTACAATCCATGCCAACCCGTTAA	MAKGWFAHSQFSARIDWLQFLAV SEVPPASIQVEVARLHAPRGGRQG IGRRLQSMPTR
150	ATGACCCAGAGCGCTGCCGGCACCTGTCCTACTTTA CCGGGTTGGACTCAAGACGATAGTTACCGGATAAG GCGCAGCGGTCGGGCTGAACGGCACACAGCCCAGC TTGGAGCGAACGACCTACACCGAACTGAGATAGGT ACAGCGTGA	MTQSAAGTCPTLPGWTQDDSYRI RRSGRAERHTAQLGANDLHRTEIG TA
138	ATGGTCGTCATCTACCTGCCTGGACAGCATGGCCTG CAACGCGGGCCGCCGGAAGCGAGAAGAATCATAA TGGGGAAGGCCATCCAGCCTCGCGTCGCGAACGCC AGCCCAGCGCGTCGGCCGCCATGCCGGCGATAA	MVVIYLPGQHGLQRGPPEARRIIM GKAIQPRVANASPARRPPCRR
93	ATGGAGCCGGGCCACCTCGACCTGAATGGAAGCCG GCGGCACCTCGCTAACGGCCAAGAATTGGAGCCAA TCAATTCTTGCGGAGAACTGTGA	MEPGHLDLNGSRRHLANGQELEPI NSCGEL
72	ATGGCCTTCCCCATTATGATTCTTCTCGCTTCCGGC GGCCCGCGTTGCAGGCCATGCTGTCCAGGCAGGTA G	MAFPIMILLASGGPRCRPCCPGR
72	ATGGATTGTAGGCGCCGCCCTATACCTTGTCTGCCT CCCCGCGGTGCATGGAGCCGGGCCACCTCGACCTG A	MDCRRRPIPCLPPRGAWSRATST

72	ATGCTCGTCAGGGGGGCGGAGCCTATGGAAAAACG CCAGCAACGCGTCCATGTGCGGCATAAATCGCCGT GA	MLVRGAEPMEKRQQRVHVRHKSP
69	ATGGGGAAGGCCATCCAGCCTCGCGTCGCGAACGC CAGCCCAGCGCGTCGGCCGCCATGCCGGCGATAA	MGKAIQPRVANASPARRPPCRR
63	ATGCTGTCCAGGCAGGTAGATGACGACCATCAGGG ACAGCTTCAACGGCTCTTACCAGCCTAA	MLSRQVDDDHQGQLQRLLPA
63	ATGGCGGCCGACGCGCTGGGCTGGCGTTCGCGACG CGAGGCTGGATGGCCTTCCCCATTATGA	MLSRQVDDDHQGQLQRLLPA
57	ATGATTCTTCTCGCTTCCGGCGGCCCGCGTTGCAGG CCATGCTGTCCAGGCAGGTAG	MILLASGGPRCRPCCPGR
51	ATGTGCGGCATAAATCGCCGTGACGATCAGCGGTC CAGTGATCGAAGTTAG	MCGINRRDDQRSSDRS
48	ATGGAAAAACGCCAGCAACGCGTCCATGTGCGGCA TAAATCGCCGTGA	MEKRQQRVHVRHKSP
45	ATGCCGGCGATAATGGCCTGCTTCTCGCCGAAACG TTTGGAGTGA	MPAIMACFSPKRLE
42	ATGGCCTGCAACGCGGGCCGCCGGAAGCGAGAAG AATCATAA	MACNAGRRKREES
33	ATGGCCTGCTTCTCGCCGAAACGTTTGGAGTGA	MACFSPKRLE
30	ATGCACCGCGGGGAGGCAGACAAGGTATAG	MHRGEADKV
15	ATGCTCACGCTGTAG	MLTL
15	ATGCCAACCCGTTAA	MPTR

Results from the ORF finder:

```python
#reading the fasta file
sequence = SeqIO.read("part2_Dinosaur_DNA.fasta", "fasta")

# Printing the sequence and its ID
print("Sequence:", sequence.seq)
print("ID:", sequence.id)

def find_all_ORFs(sequence):
    ORFs = []
    #finding reverse complement and iterating over 3 frames for both sequences
    for strand, nuc in [(+1, sequence.seq), (-1, sequence.seq.reverse_complement())]:
        for frame in range(3):
            length = 3 * ((len(sequence)-frame) // 3) # Multiple of three for codons
            for i in range(frame, length, 3):
                codon = nuc[i:i+3]
                if codon in ['ATG', 'AUG']:
                    for j in range(i+3, length, 3):
                        if nuc[j:j+3] in ['TAA', 'TAG', 'TGA', 'UAA', 'UAG', 'UGA']:
                            ORFs.append(nuc[i:j+3])
                            break
    ORFs.sort(key=len, reverse=True)
    return ORFs  # Now returning a list of all ORFs, in order of size

ORFs = find_all_ORFs(sequence)

# Create a table to input results into
table = PrettyTable()

# Adding columns to the table
table.field_names = ["ORF Length", "ORF Sequence", "Amino Acid Sequence"]

# saving ORFs as fasta files and adding the translated amino acid sequence for each ORF into the table
for i, orf in enumerate(ORFs, 1):
    amino_acid_sequence = orf.translate(to_stop=True)
    table.add_row([len(orf), orf, amino_acid_sequence])
    record = SeqRecord(orf, id=f"ORF{i}", description=f"ORF{i} from sequence {sequence.id}")
    SeqIO.write(record, f"ORF{i}.fasta", "fasta")

# Print results in the table
print(table)

# Print the number of ORFs found
print("Number of ORFs found:", len(ORFs))
```

```
+-------------------------------------
-------------------------------------+
Number of ORFs found: 24
```

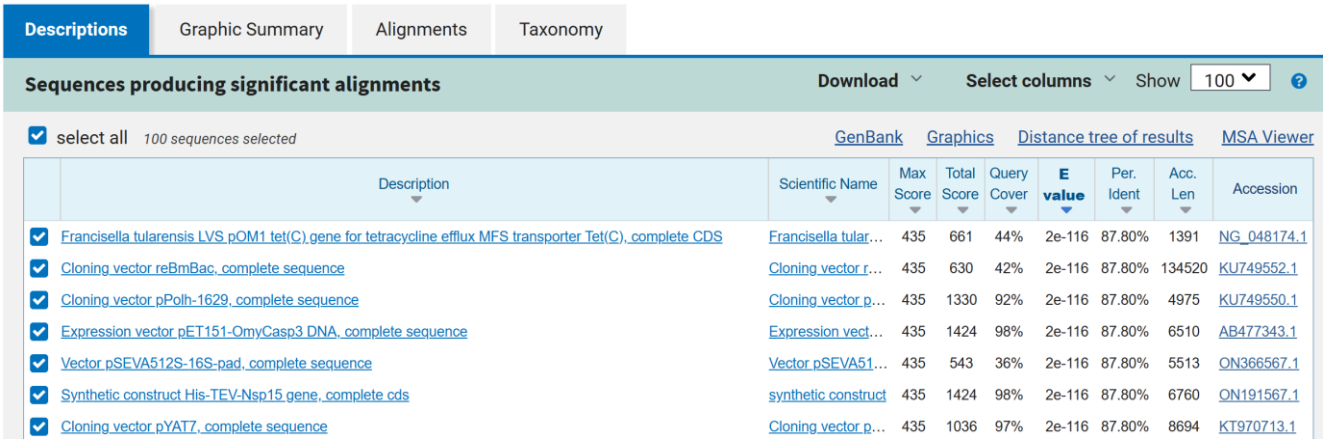
Sequence-1: 'Dinosaur' Protein analysis.

Protein type	E-value and coverage	comments
Hypothetical protein from Escherichia coli	E-value: 1e-10 Coverage: 33%	-Many repeat outputs
plasmid replication initiation protein, partial [Klebsiella pneumoniae]	E-value: 6e-10 Coverage:42%	-Many repeat outputs
Hypothetical protein from Salmonella enterica	E-value: 7e-04 Coverage: 51%	-Many repeat outputs
Hypothetical protein (VEV11M) from Escherichia coli	E-value: 5e-04 Coverage: 100%	-Many repeat outputs

Sequence-1: comments and conclusions

The high proportion of hypothetical/ partial bacterial protein sequences appears telling of a typical case of sample contamination. However, as no matches other than the repeated list of hypothetical bacterial protein sequences were returned, it seems unlikely that there was any origional sample amplified in the first place.

I decided to investigate with this unknown original 'Dinosaur' sequence further and run some BLASTn searches on the sequence and it's longest ORFs. A BLAST of the full sequence (see below) returned weak alignments of bacterial origin, solidifying to findings from the original BLASTp searches from the ORF finder output.



Descriptions | Graphic Summary | Alignments | Taxonomy

Sequences producing significant alignments — Download — Select columns — Show 100

select all *100 sequences selected* — GenBank | Graphics | Distance tree of results | MSA Viewer

Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
Francisella tularensis LVS pOM1 tet(C) gene for tetracycline efflux MFS transporter Tet(C), complete CDS	Francisella tular…	435	661	44%	2e-116	87.80%	1391	NG_048174.1
Cloning vector reBmBac, complete sequence	Cloning vector r…	435	630	42%	2e-116	87.80%	134520	KU749552.1
Cloning vector pPolh-1629, complete sequence	Cloning vector p…	435	1330	92%	2e-116	87.80%	4975	KU749550.1
Expression vector pET151-OmyCasp3 DNA, complete sequence	Expression vect…	435	1424	98%	2e-116	87.80%	6510	AB477343.1
Vector pSEVA512S-16S-pad, complete sequence	Vector pSEVA51…	435	543	36%	2e-116	87.80%	5513	ON366567.1
Synthetic construct His-TEV-Nsp15 gene, complete cds	synthetic construct	435	1424	98%	2e-116	87.80%	6760	ON191567.1
Cloning vector pYAT7, complete sequence	Cloning vector p…	435	1036	97%	2e-116	87.80%	8694	KT970713.1

BLASTn searches of the three longest ORFs for this sequence also agreed with the original BLASTp searches. These results lead me to believe that either:

1) The sequence is fictitious
2) Or, somehow through storing and processing he sample decades later, the original sample has degraded and/ or has become contaminated with bacterial DNA

In this situation I would lean towards the sequence being entirely fictitious.

Introduction – Sequence 2

In this section (Part 2, section 2) an updated sequence of possible Dinosaur DNA was provided by Mark Boguski, to improve upon the seemingly fictitious sequence from section 1 of part 2. Again, the ORF detector coded-for in Part 1 will be utilised as well as the NCBI BLAST+ online webservice.

Sequence-2: Finding 'Dinosaur' ORFs

Dinosaur DNA – Sequence 2:

```
GAATTCCGGAAGCGAGCAAGAGATAAGTCCTGGCATCAGATACAGTTGGAGATAAGGACGGACGTG
TGGCAGCTCCCGCAGAGGATTCACTGGAAGTGCATTACCTATCCCATGGGAGCCATGGAGTTCGTGG
CGCTGGGGGGGCCGGATGCGGGCTCCCCCACTCCGTTCCCTGATGAAGCCGGAGCCTTCCTGGGGC
TGGGGGGGGGCGAGAGGACGGAGGCGGGGGGGCTGCTGGCCTCCTACCCCCCCTCAGGCCGCGTG
TCCCTGGTGCCGTGGGCAGACACGGGTACTTTGGGGACCCCCCAGTGGGTGCCGCCCGCCACCCAA
ATGGAGCCCCCCCCACTACCTGGAGCTGCTGCAACCCCCCCCGGGGGCAGCCCCCCCCCATCCCTCCTCC
GGGCCCCTACTGCCACTCAGCAGCGGGCCCCCACCCTGCGAGGCCCGTGAGTGCGTCATGGCCAGG
AAGAACTGCGGAGCGACGGCAACGCCGCTGTGGCGCCGGGACGGCACGGGCATTACCTGTGCAAC
TGGGCCTCAGCCTGCGGGCTCTACCACCGCCTCAACGGCCAGAACCGCCCGCTCATCCGCCCCAAA
AAGCGCCTGCGGGTGAGTAAGCGCGCAGGCACAGTGTGCAGCCACGAGCGTGAAAACTGCCAGAC
ATCCACCACCACTCTGTGGCGTCGCAGCCCCATGGGGGACCCCGTCTGCAACAACATTCACGCCTG
CGGCCTCTACTACAAACTGCACCAAGTGAACCGCCCCCTCACGATGCGCAAAGACGGAATCCAAAC
```

CCGAAACCGCAAAGTTTCCTCCAAGGGTAAAAAGCGGCGCCCCCCGGGGGGGGGGAAACCCCTCCG
CCACCGCGGGAGGGGGCGCTCCTATGGGGGGAGGGGGGGACCCCTCTATGCCCCCCCCCGCCGCCCC
CCCCGGCCGCCGCCCCCCCTCAAAGCGACGCTCTGTACGCTCTCGGCCCCGTGGTCCTTTCGGGCC
ATTTTCTGCCCTTTGGAAACTCCGGAGGGTTTTTTGGGGGGGGGGCGGGGGGTTACACGGCCCCCC
CGGGGCTGAGCCCGCAGATTTAAATAATAACTCTGACGTGGGCAAGTGGGCCTTGCTGAGAAGACA
GTGTAACATAATAATTTGCACCTCGGCAATTGCAGAGGGTCGATCTCCACTTTGGACACAACAGGGC
TACTCGGTAGGACCAGATAAGCACTTTGCTCCCTGGACTGAAAAAGAAAGGATTTATCTGTTTGCTT
CTTGCTGACAAATCCCTGTGAAAGGTAAAAGTCGGACACAGCAATCGATTATTTCTCGCCTGTGTGA
AATTACTGTGAATATTGTAAATATATATATATATATATATATATCTGTATAGAACAGCCTCGGAGGCGGCA
TGGACCCAGCGTAGATCATGCTGGATTTGTACTGCCGGAATTC

Results from the ORF finder:

ORF Length	ORF Sequence	Amino Acid Sequence
966	ATGGGAGCCATGGAGTTCGTGGCGCTGGGGGGG CCGGATGCGGGCTCCCCCACTCCGTTCCCTGATG AAGCCGGAGCCTTCCTGGGGCTGGGGGGGGGGCG AGAGGACGGAGGCGGGGGGGCTGCTGGCCTCCT ACCCCCCCTCAGGCCGCGTGTCCCTGGTGCCGT GGGCAGACACGGGTACTTTGGGGACCCCCCAGT GGGTGCCGCCCGCCACCCAAATGGAGCCCCCCC ACTACCTGGAGCTGCTGCAACCCCCCCCGGGGCA GCCCCCCCCCATCCCTCCTCCGGGCCCCTACTGCC ACTCAGCAGCGGGCCCCCACCCTGCGAGGCCCG TGAGTGCGTCATGGCCAGGAAGAACTGCGGAGC GACGGCAACGCCGCTGTGGCGCCGGGACGGCAC CGGGCATTACCTGTGCAACTGGGCCTCAGCCTGC GGGCTCTACCACCGCCTCAACGGCCAGAACCGC CCGCTCATCCGCCCCAAAAAGCGCCTGCGGGTG AGTAAGCGCGCAGGCACAGTGTGCAGCCACGAG CGTGAAAACTGCCAGACATCCACCACCACTCTG TGGCGTCGCAGCCCCATGGGGGACCCCGTCTGC AACAACATTCACGCCTGCGGCCTCTACTACAAAC TGCACCAAGTGAACCGCCCCCTCACGATGCGCA AAGACGGAATCCAAACCCGAAACCGCAAAGTTT CCTCCAAGGGTAAAAAGCGGCGCCCCCCGGGGG GGGGAAACCCCTCCGCCACCGCGGGAGGGGGC GCTCCTATGGGGGGAGGGGGGGACCCCTCTATG CCCCCCCCCGCCGCCCCCCCCGGCCGCCGCCCCC CCTCAAAGCGACGCTCTGTACGCTCTCGGCCCC GTGGTCCTTTCGGGCCATTTTCTGCCCTTTGGAA ACTCCGGAGGGTTTTTTGGGGGGGGGGCGGGGG GTTACACGGCCCCCCCGGGGCTGAGCCCGCAGA TTTAA	MGAMEFVALGGPDAGS PTPFPDEAGAFLGLGGG ERTEAGGLLASYPPSGR VSLVPWADTGTLGTPQ WVPPATQMEPPHYLELL QPPRGSPPHPSSGPLLPL SSGPPPCEARECVMARK NCGATATPLWRRDGTGH YLCNWASACGLYHRLN GQNRPLIRPKKRLRVSK RAGTVCSHERENCQTST TTLWRRSPMGDPVCNNI HACGLYYKLHQVNRPLT MRKDGIQTRNRKVSSK GKKRRPPGGGNPSATAG GGAPMGGGGDPSMPPP PPPPAAAPPQSDALYALG PVVLSGHFLPFGNSGGF FGGGAGGYTAPPGLSPQ I
957	ATGGAGTTCGTGGCGCTGGGGGGGCCGGATGCG GGCTCCCCCACTCCGTTCCCTGATGAAGCCGGA GCCTTCCTGGGGCTGGGGGGGGGCGAGAGGAC GGAGGCGGGGGGGCTGCTGGCCTCCTACCCCCC CTCAGGCCGCGTGTCCCTGGTGCCGTGGGCAGA CACGGGTACTTTGGGGACCCCCCAGTGGGTGCC GCCCGCCACCCAAATGGAGCCCCCCCACTACCT GGAGCTGCTGCAACCCCCCCCGGGGCAGCCCCCC CCATCCCTCCTCCGGGCCCCTACTGCCACTCAGC AGCGGGCCCCCACCCTGCGAGGCCCGTGAGTGC GTCATGGCCAGGAAGAACTGCGGAGCGACGGCA ACGCCGCTGTGGCGCCGGGACGGCACCGGGCAT	MEFVALGGPDAGSPTPF PDEAGAFLGLGGGERTE AGGLLASYPPSGRVSLV PWADTGTLGTPQWVPPA TQMEPPHYLELLQPPRG SPPHPSSGPLLPLSSGPPP CEARECVMARKNCGAT ATPLWRRDGTGHYLCN WASACGLYHRLNGQNR PLIRPKKRLRVSKRAGT VCSHERENCQTSTTTLW RRSPMGDPVCNNIHACG

	TACCTGTGCAACTGGGCCTCAGCCTGCGGGCTCT ACCACCGCCTCAACGGCCAGAACCGCCCGCTCA TCCGCCCCAAAAAGCGCCTGCGGGTGAGTAAGC GCGCAGGCACAGTGTGCAGCCACGAGCGTGAA AACTGCCAGACATCCACCACCACTCTGTGGCGT CGCAGCCCCATGGGGGACCCCGTCTGCAACAAC ATTCACGCCTGCGGCCTCTACTACAAACTGCACC AAGTGAACCGCCCCCTCACGATGCGCAAAGACG GAATCCAAACCCGAAACCGCAAAGTTTCCTCCA AGGGTAAAAAGCGGCGCCCCCCGGGGGGGGGA AACCCCTCCGCCACCGCGGGAGGGGGCGCTCCT ATGGGGGGAGGGGGGGACCCCTCTATGCCCCCC CCGCCGCCCCCCCCCGGCCGCCGCCCCCCCTCAA AGCGACGCTCTGTACGCTCTCGGCCCCGTGGTCC TTTCGGGCCATTTTCTGCCCTTTGGAAACTCCGG AGGGTTTTTTGGGGGGGGGGCGGGGGGTTACAC GGCCCCCCCCGGGGCTGAGCCCGCAGATTTAA	LYYKLHQVNRPLTMRK DGIQTRNRKVSSKGKKR RPPGGGNPSATAGGGAP MGGGGDPSMPPPPPPPA AAPPQSDALYALGPVVL SGHFLPFGNSGGFFGGG AGGYTAPPGLSPQI
747	ATGGAGCCCCCCCACTACCTGGAGCTGCTGCAA CCCCCCCGGGGCAGCCCCCCCCATCCCTCCTCCG GGCCCCTACTGCCACTCAGCAGCGGGCCCCCAC CCTGCGAGGCCCGTGAGTGCGTCATGGCCAGGA AGAACTGCGGAGCGACGGCAACGCCGCTGTGGC GCCGGGACGGCACCGGGCATTACCTGTGCAACT GGGCCTCAGCCTGCGGGCTCTACCACCGCCTCA ACGGCCAGAACCGCCCGCTCATCCGCCCCAAAA AGCGCCTGCGGGTGAGTAAGCGCGCAGGCACAG TGTGCAGCCACGAGCGTGAAAACTGCCAGACAT CCACCACCACTCTGTGGCGTCGCAGCCCCATGG GGGACCCCGTCTGCAACAACATTCACGCCTGCG GCCTCTACTACAAACTGCACCAAGTGAACCGCC CCCTCACGATGCGCAAAGACGGAATCCAAACCC GAAACCGCAAAGTTTCCTCCAAGGGTAAAAAGC GGCGCCCCCCGGGGGGGGGAAACCCCTCCGCCA CCGCGGGAGGGGGCGCTCCTATGGGGGGAGGGG GGGACCCCTCTATGCCCCCCCCGCCGCCCCCCCC GGCCGCCGCCCCCCCCTCAAAGCGACGCTCTGTA CGCTCTCGGCCCCGTGGTCCTTTCGGGCCATTTT CTGCCCTTTGGAAACTCCGGAGGGTTTTTTGGGG GGGGGGCGGGGGGTTACACGGCCCCCCCGGGGC TGAGCCCGCAGATTTAA	MEPPHYLELLQPPRGSP PHPSSGPLLPLSSGPPPC EARECVMARKNCGATA TPLWRRDGTGHYLCNW ASACGLYHRLNGQNRPL IRPKKRLRVSKRAGTVC SHERENCQTSTTTLWRR SPMGDPVCNNIHACGLY YKLHQVNRPLTMRKDGI QTRNRKVSSKGKKRRPP GGGNPSATAGGGAPMG GGGDPSMPPPPPPPAAA PPQSDALYALGPVVLSG HFLPFGNSGGFFGGGAG GYTAPPGLSPQI
624	ATGGCCAGGAAGAACTGCGGAGCGACGGCAAC GCCGCTGTGGCGCCGGGACGGCACCGGGCATTA CCTGTGCAACTGGGCCTCAGCCTGCGGGCTCTA CCACCGCCTCAACGGCCAGAACCGCCCGCTCAT CCGCCCCAAAAAGCGCCTGCGGGTGAGTAAGCG CGCAGGCACAGTGTGCAGCCACGAGCGTGAAA ACTGCCAGACATCCACCACCACTCTGTGGCGTC GCAGCCCCATGGGGGACCCCGTCTGCAACAACA TTCACGCCTGCGGCCTCTACTACAAACTGCACCA AGTGAACCGCCCCCTCACGATGCGCAAAGACGG AATCCAAACCCGAAACCGCAAAGTTTCCTCCAA GGGTAAAAAGCGGCGCCCCCCGGGGGGGGGAA ACCCCTCCGCCACCGCGGGAGGGGGCGCTCCTA TGGGGGGAGGGGGGGGACCCCTCTATGCCCCCCC CGCCGCCCCCCCCCGGCCGCCGCCCCCCCTCAAA GCGACGCTCTGTACGCTCTCGGCCCCGTGGTCCT TTCGGGCCATTTTCTGCCCTTTGGAAACTCCGGA GGGTTTTTTGGGGGGGGGGCGGGGGGTTACACG GCCCCCCCGGGGCTGAGCCCGCAGATTTAA	MARKNCGATATPLWRR DGTGHYLCNWASACGL YHRLNGQNRPLIRPKKR LRVSKRAGTVCSHEREN CQTSTTTLWRRSPMGDP VCNNIHACGLYYKLHQ VNRPLTMRKDGIQTRNR KVSSKGKKRRPPGGGNP SATAGGGAPMGGGGDP SMPPPPPPPAAAPPQSDA LYALGPVVLSGHFLPFG NSGGFFGGGAGGYTAPP GLSPQI

462	ATGCGGGCTCCCCCACTCCGTTCCCTGATGAAGC CGGAGCCTTCCTGGGGCTGGGGGGGGGCGAGA GGACGGAGGCGGGGGGGCTGCTGGCCTCCTACC CCCCCTCAGGCCGCGTGTCCCTGGTGCCGTGGG CAGACACGGGTACTTTGGGGACCCCCCAGTGGG TGCCGCCCGCCACCCAAATGGAGCCCCCCCACT ACCTGGAGCTGCTGCAACCCCCCCGGGGCAGCC CCCCCCATCCCTCCTCCGGGCCCCTACTGCCACT CAGCAGCGGGCCCCCACCCTGCGAGGCCCGTGA GTGCGTCATGGCCAGGAAGAACTGCGGAGCGAC GGCAACGCCGCTGTGGCGCCGGGACGGCACCGG GCATTACCTGTGCAACTGGGCCTCAGCCTGCGG GCTCTACCACCGCCTCAACGGCCAGAACCGCCC GCTCATCCGCCCCAAAAAGCGCCTGCGGGTGA	MRAPPLRSLMKPEPSW GWGGARGRRRGGCWP PTPPQAACPWCRGQTRV LWGPPSGCRPPPKWSPP TTWSCCNPPGAAPPIPPP GPYCHSAAGPHPARPVS ASWPGRTAERRQRRCG AGTAPGITCATGPQPAGS TTASTARTARSSAPKSAC G
456	ATGGCCCGAAAGGACCACGGGGCCGAGAGCGTA CAGAGCGTCGCTTTGAGGGGGGGCGGCGGCCGG GGGGGGCGGCGGGGGGGGCATAGAGGGGTCCC CCCCTCCCCCCATAGGAGCGCCCCCTCCCGCGGT GGCGGAGGGGTTTCCCCCCCCCGGGGGGCGCCG CTTTTTACCCTTGGAGGAAACTTTGCGGTTTCGG GTTTGGATTCCGTCTTTGCGCATCGTGAGGGGGC GGTTCACTTGGTGCAGTTTGTAGTAGAGGCCGCA GGCGTGAATGTTGTTGCAGACGGGGTCCCCCAT GGGGCTGCGACGCCACAGAGTGGTGGTGGATGT CTGGCAGTTTTCACGCTCGTGGCTGCACACTGTG CCTGCGCGCTTACTCACCCGCAGGCGCTTTTTGG GGCGGATGAGCGGGCGGTTCTGGCCGTTGAGGC GGTGGTAGAGCCCGCAGGCTGA	MARKDHGAESVQSVAL RGGGGRGGRRGGHRGV PPSPHRSAPSRGGGGVS PPRGAPLFTLGGNFAVS GLDSVFAHREGAVHLVQ FVVEAAGVNVVADGVP HGAATPQSGGGCLAVFT LVAAHCACALTHPQALF GADERAVLAVEAVVEPA G
435	ATGAAGCCGGAGCCTTCCTGGGGCTGGGGGGGG GCGAGAGGACGGAGGCGGGGGGGCTGCTGGCC TCCTACCCCCCCTCAGGCCGCGTGTCCCTGGTGC CGTGGGCAGACACGGGTACTTTGGGGACCCCCC AGTGGGTGCCGCCCGCCACCCAAATGGAGCCCC CCCACTACCTGGAGCTGCTGCAACCCCCCCGGG GCAGCCCCCCCCATCCCTCCTCCGGGCCCCTACT GCCACTCAGCAGCGGGCCCCCACCCTGCGAGGC CCGTGAGTGCGTCATGGCCAGGAAGAACTGCGG AGCGACGGCAACGCCGCTGTGGCGCCGGGACG GCACCGGGCATTACCTGTGCAACTGGGCCTCAG CCTGCGGGCTCTACCACCGCCTCAACGGCCAGA ACCGCCCGCTCATCCGCCCCAAAAAGCGCCTGC GGGTGA	MKPEPSWGWGGARGR RRGGCWPPTPPQAACP WCRGQTRVLWGPPSGC RPPPKWSPPTTWSCCNP PGAAPPIPPPGPYCHSAA GPHPARPVSASWPGRTA ERRQRRCGAGTAPGITC ATGPQPAGSTTASTARTA RSSAPKSACG
387	ATGGGGGACCCCGTCTGCAACAACATTCACGCC TGCGGCCTCTACTACAAACTGCACCAAGTGAAC CGCCCCCTCACGATGCGCAAAGACGGAATCCAA ACCCGAAACCGCAAAGTTTCCTCCAAGGGTAAA AAGCGGCGCCCCCCGGGGGGGGGGAAACCCCTCC GCCACCGCGGGAGGGGGCGCTCCTATGGGGGGA GGGGGGGACCCCTCTATGCCCCCCCCGCCGCCC CCCCCGGCCGCCGCCCCCCCTCAAAGCGACGCT CTGTACGCTCTCGGCCCCGTGGTCCTTTCGGGCC ATTTTCTGCCCTTTGGAAACTCCGGAGGGTTTTT TGGGGGGGGGGCGGGGGGTTACACGGCCCCCCC GGGGCTGAGCCCGCAGATTTAA	MGDPVCNNIHACGLYY KLHQVNRPLTMRKDGIQ TRNRKVSSKGKKRRPPG GGNPSATAGGGAPMGG GGDPSMPPPPPPPAAAPP QSDALYALGPVVLSGHF LPFGNSGGFFGGGAGGY TAPPGLSPQI
309	ATGCGCAAAGACGGAATCCAAACCCGAAACCGC AAAGTTTCCTCCAAGGGTAAAAAGCGGCGCCCC CCGGGGGGGGGAAACCCCTCCGCCACCGCGGG AGGGGGCGCTCCTATGGGGGGAGGGGGGGACCC CTCTATGCCCCCCCCGCCGCCCCCCCCGGCCGCC	MRKDGIQTRNRKVSSK GKKRRPPGGGNPSATAG GGAPMGGGGDPSMPPP PPPPAAAPPQSDALYALG PVVLSGHFLPFGNSGGF

	GCCCCCCCTCAAAGCGACGCTCTGTACGCTCTCG GCCCCGTGGTCCTTTCGGGCCATTTTCTGCCCTT TGGAAACTCCGGAGGGTTTTTTGGGGGGGGGGC GGGGGGTTACACGGCCCCCCCGGGGCTGAGCCC GCAGATTTAA	FGGGAGGYTAPPGLSPQ I
228	ATGTTACACTGTCTTCTCAGCAAGGCCCACTTGC CCACGTCAGAGTTATTATTTAAATCTGCGGGCTC AGCCCCGGGGGGGCCGTGTAACCCCCCGCCCCC CCCCCAAAAAACCCTCCGGAGTTTCCAAAGGGC AGAAAATGGCCCGAAAGGACCACGGGGCCGAG AGCGTACAGAGCGTCGCTTTGAGGGGGGGCGGC GGCCGGGGGGGGCGGCGGGGGGGGCATAG	MLHCLLSKAHLPTSELL FKSAGSAPGGPCNPPPPP QKTLRSFQRAENGPKGP RGRERTERRFEGGRRPG GAAGGA
198	ATGGGGGGAGGGGGGGACCCCTCTATGCCCCCC CCGCCGCCCCCCCCCGGCCGCCGCCCCCCCTCAA AGCGACGCTCTGTACGCTCTCGGCCCCGTGGTCC TTTCGGGCCATTTTCTGCCCTTTGGAAACTCCGG AGGGTTTTTTGGGGGGGGGGCGGGGGGTTACAC GGCCCCCCCCGGGGCTGAGCCCGCAGATTTAA	MGGGGDPSMPPPPPPPA AAPPQSDALYALGPVVL SGHFLPFGNSGGFFGGG AGGYTAPPGLSPQI
174	ATGCCCCCCCCCGCCGCCCCCCCCCGGCCGCCGCCC CCCCTCAAAGCGACGCTCTGTACGCTCTCGGCCC CGTGGTCCTTTCGGGCCATTTTCTGCCCTTTGGA AACTCCGGAGGGTTTTTTGGGGGGGGGGCGGGG GGTTACACGGCCCCCCCCGGGGCTGAGCCCGCAG ATTTA	MPPPPPPPAAAPPQSDAL YALGPVVLSGHFLPFGN SGGFFGGGAGGYTAPPG LSPQI
171	ATGATCTACGCTGGGTCCATGCCGCCTCCGAGGC TGTTCTATACAGATATATATATATATATATATATATTT ACAATATTCACAGTAATTTCACACAGGCGAGAAA TAATCGATTGCTGTGTCCGACTTTTACCTTTCACA GGGATTTGTCAGCAAGAAGCAAACAGATAA	MIYAGSMPPPRLFYTDIY IYIYIFTIFTVISHRREIID CCVRLLPFTGICQQEAN R
168	ATGTTGTTGCAGACGGGGTCCCCCATGGGGCTGC GACGCCACAGAGTGGTGGTGGATGTCTGGCAGT TTTCACGCTCGTGGCTGCACACTGTGCCTGCGCG CTTACTCACCCGCAGGCGCTTTTTGGGGCGGATG AGCGGGCGGTTCTGGCCGTTGAGGCGGTGGTAG	MLLQTGSPMGLRRHRV VVDVWQFSRSWLHTVP ARLLTRRRFLGRMSGRF WPLRRW
153	ATGCCGCCTCCGAGGCTGTTCTATACAGATATATA TATATATATATATATATTTACAATATTCACAGTAATT TCACACAGGCGAGAAATAATCGATTGCTGTGTCC GACTTTTACCTTTCACAGGGATTTGTCAGCAAGA AGCAAACAGATAA	MPPPRLFYTDIYIYIYIFT IFTVISHRREIIDCCVRLL PFTGICQQEANR
144	ATGGGGCTGCGACGCCACAGAGTGGTGGTGGAT GTCTGGCAGTTTTCACGCTCGTGGCTGCACACTG TGCCTGCGCGCTTACTCACCCGCAGGCGCTTTTT GGGGCGGATGAGCGGGCGGTTCTGGCCGTTGAG GCGGTGGTAG	MGLRRHRVVVDVWQFS RSWLHTVPARLLTRRRF LGRMSGRFWPLRRW
135	ATGGGGGGGGCTGCCCCGGGGGGGTTGCAGCAG CTCCAGGTAGTGGGGGGGCTCCATTTGGGTGGC GGGCGGCACCCACTGGGGGGTCCCCAAAGTACC CGTGTCTGCCCACGGCACCAGGGACACGCGGCC TGA	MGGAAPGGLQQLQVVG GLHLGGGRHPLGGPQST RVCPRHQGHAA
114	ATGCCCGGTGCCGTCCCGGCGCCACAGCGGCGT TGCCGTCGCTCCGCAGTTCTTCCTGGCCATGACG CACTCACGGGCCTCGCAGGGTGGGGGCCCGCTG CTGAGTGGCAGTAG	MPGAVPAPQRRCRRSAV LPGHDALTGLAGWGPA AEWQ
111	ATGACGCACTCACGGGCCTCGCAGGGTGGGGGC CCGCTGCTGAGTGGCAGTAGGGGCCCGGAGGAG GGATGGGGGGGGCTGCCCCGGGGGGGTTGCAGC	MTHSRASQGGGPLLSGS RGPEEGWGGLPRGGCSS SR

	AGCTCCAGGTAG	
81	ATGTCTGGCAGTTTTCACGCTCGTGGCTGCACACTGTGCCTGCGCGCTTACTCACCCGCAGGCGCTTTTTGGGGCGGATGA	MSGSFHARGCTLCLRAYSPAGAFWGG
36	ATGAGCGGGCGGTTCTGGCCGTTGAGGCGGTGGTAG	MSGRFWPLRRW
18	ATGGCTCCCATGGGATAG	MAPMG
15	ATGGACCCAGCGTAG	MDPA
15	ATGCACTTCCAGTGA	MHFQ
9	ATGGGATAG	MG

Comments:

More ORFs were detected in the updated sequence than with the original 'Dinosaur' DNA sequence in Part 2:Section1. The ORFs returned by the ORD detector are also longer in length with this updated sequence, which could provide more information to the sequence, as long ORFs are often used to aid in initially identifying candidate protein-coding regions or functional RNA-coding regions in a DNA sequence. (Deonier et.al, 2005) However, this observation alone is not enough to suggest that the long reading frames are conclusive enough evidence for the presence of protein coding genes within the new sequence.

Code input:

```
#reading the fasta file
sequence = SeqIO.read("part2_Dino_DNA_v2.fasta", "fasta")

# Printing the sequence and its ID
print("Sequence:", sequence.seq)
print("ID:", sequence.id)

def find_all_ORFs(sequence):
    ORFs = []
    #finding reverse complement and iterating over 3 frames for both sequences
    for strand, nuc in [(+1, sequence.seq), (-1, sequence.seq.reverse_complement())]:
        for frame in range(3):
            length = 3 * ((len(sequence)-frame) // 3) # Multiple of three for codons
            for i in range(frame, length, 3):
                codon = nuc[i:i+3]
                if codon in ['ATG', 'AUG']:
                    for j in range(i+3, length, 3):
                        if nuc[j:j+3] in ['TAA', 'TAG', 'TGA', 'UAA', 'UAG', 'UGA']:
                            ORFs.append(nuc[i:j+3])
                            break
    ORFs.sort(key=len, reverse=True)
    return ORFs  # Now returning a list of all ORFs, in order of size

ORFs = find_all_ORFs(sequence)

# Create a table to input results into
table = PrettyTable()

# Adding columns to the table
table.field_names = ["ORF Length", "ORF Sequence", "Amino Acid Sequence"]

# saving ORFs as fasta files and adding the translated amino acid sequence for each ORF into the table
for i, orf in enumerate(ORFs, 1):
    amino_acid_sequence = orf.translate(to_stop=True)
    table.add_row([len(orf), orf, amino_acid_sequence])
    record = SeqRecord(orf, id=f"ORF{i}", description=f"ORF{i} from sequence {sequence.id}")
    SeqIO.write(record, f"ORF{i}.fasta", "fasta")

# Print results in the table
print(table)

# Print the number of ORFs found
print("Number of ORFs found:", len(ORFs))
```

```
----------------------------
----------------------------
Number of ORFs found: 25
```

Sequence-2: 'Dinosaur' Protein analysis

Protein type	E-value and coverage	Comments
Erythroid transcription factor from Gallus *gallus*	E-value: 0 Coverage: 99%	-protein from the longest ORF -Gallus *gallus* = domestic chicken -Multiple hits for the same TF

Sequence-2: comments and conclusions

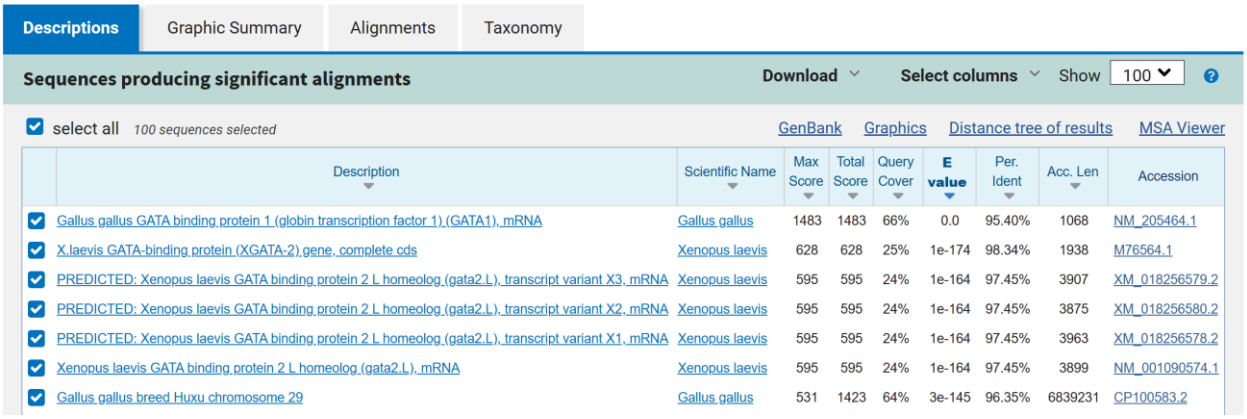
The ORF finder output provided a lot of reasonably long protein sequences which come up as unknown on BLAST search, along with repetition after repetition of the sequence for Erythroid transcription factor from Gallus *gallus* in the longest sequence outputs.

First, like every good researcher, I will humour all possible explanation as to the origin of our second 'Dinosaur' sequence. Our current output from BLASTp is the amio acid sequence for the Erythroid transcription factor (of chicken origin), and a list of completely unrecognised sequences. According to Feduccia Alan in the journal of BioScience (Alan et.al, pages 991-994, 2021) it is well established that chickens are the closest living relatives of the theropod dinosaurs today. Therefore, it would not be unreasonable to consider whether or not this sequence could be from theropod-derived DNA. Infact, A Harvard University study led by John Asara found that Trex collagen (from fossil records) was more similar to bird collagen than any other group of animals tested; which links well to a comparative analysis of the chicken genome by the International Chicken Genome Sequencing Consortium, published in 2004 (ICGSC, Nature432, 2004), suggesting that many coding regions of the chicken genome, although shortened or interspaced with more C/G repeats over the course of evolution, are fairly unchanged. So, if chicken collagen is similar to T-rex collagen, then perhaps fibrillar collagen in chicken erythocytes could be highly similar to fibrillar collagen in T-rex/later therapod erythrocyres, and therefore maybe their erythroid transcription factors could be similar in structure too?

This seems unlikely.

Whilst it is true that, as we do not have Dinosaur DNA on file we would expect a few of our amino acid sequences to, perhaps, not return a match. And, as such, if we were (hypothetically) to be given one of the first complete sections of Dinosaur DNA to be discovered, in a Cambridge University IB MCB mini project, it would be possible to argue that what we are dealing with could potentially be Dinosaur DNA. I however have decided that the aforementioned reality is incredibly silly. Instead of jumping to such conclusions, I input the entire version 2 of the mysterious 'Dinosaur' nucleotide sequence into a BLASTn search.

This was my output:



Descriptions | Graphic Summary | Alignments | Taxonomy

Sequences producing significant alignments — Download — Select columns — Show 100

select all *100 sequences selected* — GenBank — Graphics — Distance tree of results — MSA Viewer

Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
Gallus gallus GATA binding protein 1 (globin transcription factor 1) (GATA1), mRNA	Gallus gallus	1483	1483	66%	0.0	95.40%	1068	NM_205464.1
X.laevis GATA-binding protein (XGATA-2) gene, complete cds	Xenopus laevis	628	628	25%	1e-174	98.34%	1938	M76564.1
PREDICTED: Xenopus laevis GATA binding protein 2 L homeolog (gata2.L), transcript variant X3, mRNA	Xenopus laevis	595	595	24%	1e-164	97.45%	3907	XM_018256579.2
PREDICTED: Xenopus laevis GATA binding protein 2 L homeolog (gata2.L), transcript variant X2, mRNA	Xenopus laevis	595	595	24%	1e-164	97.45%	3875	XM_018256580.2
PREDICTED: Xenopus laevis GATA binding protein 2 L homeolog (gata2.L), transcript variant X1, mRNA	Xenopus laevis	595	595	24%	1e-164	97.45%	3963	XM_018256578.2
Xenopus laevis GATA binding protein 2 L homeolog (gata2.L), mRNA	Xenopus laevis	595	595	24%	1e-164	97.45%	3899	NM_001090574.1
Gallus gallus breed Huxu chromosome 29	Gallus gallus	531	1423	64%	3e-145	96.35%	6839231	CP100583.2

It is becoming more and more possible that, despite the broad dinosaur-themed claims, what we are really dealing with is a bunch of junk, non-peptide-coding DNA combined with the mRNA sequence of the Gallus *gallus* GATA binding protein 1. How anticlimactic.

Sadly, this sequence again appears to be of fictitious origins, and again is not comprehensively convincing enough to be Dinosaur DNA

Part 3 - Olfactory receptor Hunting in Mammalian Genomes

Introduction:

In Part 3 of this project, we were tasked with estimate the number and diversity of olfactory receptors in a group of mammals. Specifically, I chose to explore those of Humans, mice, cats, dolphins and dog (specifically that of a Great Dane).

Searching for olfactory protein receptors:

```
>detected_olfactory_protein
MEKRNLTVVREFVLLGLPSSAEQQHLLSVLFLCMYLATTLGNMLIIATIGFDSHLHSPMYFF
LSNLAFVDICFTSTTVPQMVVNILTGTKTISFAGCLTQLFFFVSFVNMDSLLLCVMAYDRYV
AICHPLHYTARMNLCLCVQLVAGLWLVTYLHALLHTVLIAQLSFCASNIIHHFFCDLNPLLQ
LSCSDVSFNVMIIFAVGGLLALTPLVCILVSYGLIFSTVLKITSTQGKQRAVSTCSCHLSVVVL
FYGTAIAVYFSPSSPHMPESDTLSTIMYSMVAPMLNPFIYTLRNRDMKRG
LQKMLLKCTVFQQQ
```

Python code to perform the BLAST search:

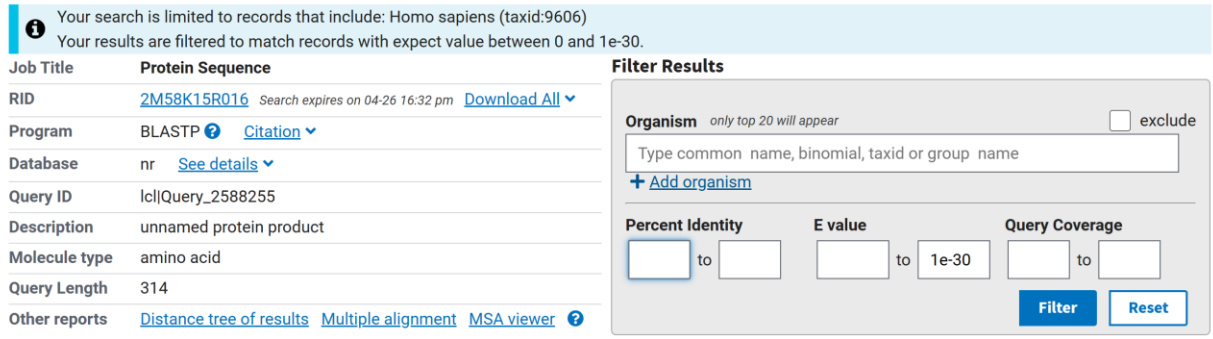
```python
# Part 3
# Importing packages
from Bio import SeqIO
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
import subprocess
from Bio.Blast import NCBIXML

# Creating a BLAST database from the Homo_sapiens.GRCh38.pep.all.fa
subprocess.run(["makeblastdb", "-in", "Homo_sapiens.GRCh38.pep.all.fa", "-dbtype", "prot"])

# Performing a BLAST search
result = subprocess.run([
    "blastp",
    "-query", "detected_olfactory_protein.fasta",
    "-db", "Homo_sapiens.GRCh38.pep.all.fa",
    "-outfmt", "5",
    "-out", "blast_results.txt"
])

# Code to check if the BLAST search ran successfully
if result.returncode != 0:
    print("Error running BLAST search")
else:
    # Parse the BLAST results
    blast_record = NCBIXML.read(open("blast_results.txt"))

# Applying filters to filter the results
filtered_results = []
for alignment in blast_record.alignments:
    for hsp in alignment.hsps:
        # a) I need to check that each hit is significant (E-value <= 1e-30)
        # b) I need to check that the % similarity is >60%
        # c) I need to make sure that the alignment is close to full length e.g. > 290 amino acids
        if hsp.expect <= 1e-30 and hsp.identities / hsp.align_length >= 0.6 and hsp.align_length >= 290:
            # d) Now, recording which matched sequences (in the Homo_sapiens.GRCh38.pep.all.fa database) satisfy a), b), c)
            filtered_results.append(alignment)

# e) Trying to work out the most likely human gene corresponding to the provided sequence
# Printing the titles of the filtered alignments, including the gene name
for alignment in filtered_results:
    print(alignment.title)

# Sanity checks:
 #Open a file in write mode
with open("filtered_results.txt", "w") as f:
    # Write the filtered results to the file
    for alignment in filtered_results:
        f.write(alignment.title + "\n")

# Print the filtered results
# Open the file in read mode
with open("filtered_results.txt", "r") as f:
    # Print the contents of the file
    print(f.read())
```

Due to system issues, I had to run the BLAST search in vscode using subroutines (see above), however this was my filtered sequence output:

```
>gnl|BL_ORD_ID|103003 ENSP00000386138.2 pep
chromosome:GRCh38:1:247757374:247758680:-1 gene:ENSG00000221888.4 transcript:ENST00000408896.4
gene_biotype:protein_coding transcript_biotype:protein_coding gene_symbol:OR1C1
description:olfactory receptor family 1 subfamily C member 1 [Source:HGNC Symbol;Acc:HGNC:8182]

MEKRNLTVVREFVLLGLPSSAEQQHLLSVLFLCMYLATTLGNMLIIATIGFDSHLHSPMY
FFLSNLAFVDICFTSTTVPQMVVNILTGTKTISFAGCLTQLFFFVSFVNMDSLLLCVMAY
DRYVAICHPLHYTARMNLCLCVQLVAGLWLVTYLHALLHTVLIAQLSFCASNIIHHFFCD
LNPLLQLSCSDVSFNVMIIFAVGGLLALTPLVCILVSYGLIFSTVLKITSTQGKQRAVST
CSCHLSVVVLFYGTAIAVYFSPSSPHMPESDTLSTIMYSMVAPMLNPFIYTLRNRDMKRG
LQKMLLKCTVFQQQ


>gnl|BL_ORD_ID|103002 ENSP00000493457.1 pep
chromosome:GRCh38:1:247754846:247760556:-1 gene:ENSG00000221888.4 transcript:ENST00000641256.1
gene_biotype:protein_coding transcript_biotype:protein_coding gene_symbol:OR1C1
description:olfactory receptor family 1 subfamily C member 1 [Source:HGNC Symbol;Acc:HGNC:8182]

MEKRNLTVVREFVLLGLPSSAEQQHLLSVLFLCMYLATTLGNMLIIATIGFDSHLHSPMY
FFLSNLAFVDICFTSTTVPQMVVNILTGTKTISFAGCLTQLFFFVSFVNMDSLLLCVMAY
DRYVAICHPLHYTARMNLCLCVQLVAGLWLVTYLHALLHTVLIAQLSFCASNIIHHFFCD
LNPLLQLSCSDVSFNVMIIFAVGGLLALTPLVCILVSYGLIFSTVLKITSTQGKQRAVST
CSCHLSVVVLFYGTAIAVYFSPSSPHMPESDTLSTIMYSMVAPMLNPFIYTLRNRDMKRG
LQKMLLKCTVFQQQ


>gnl|BL_ORD_ID|106110 ENSP00000305424.2 pep
chromosome:GRCh38:16:3188204:3206556:1 gene:ENSG00000168124.3 transcript:ENST00000304646.3
gene_biotype:protein_coding transcript_biotype:protein_coding gene_symbol:OR1F1
description:olfactory receptor family 1 subfamily F member 1 [Source:HGNC Symbol;Acc:HGNC:8194]

MSGTNQSSVSEFLLLGLSRQPQQQHLLFVFFLSMYLATVLGNLLIILSVSIDSCLHTPMY
FFLSNLSFVDICFSFTTVPKMLANHILETQTISFCGCLTQMYFVFMFVDMDNFLLAVMAY
DHFVAVCHPLHYTAKMTHQLCALLVAGLWVVANLNVLLHTLLMAPLSFCADNAITHFFCD
VTPLLKLSCSDTHLNEVIILSEGALVMITPFLCILASYMHITCTVLKVPSTKGRWKAFST
CGSHLAVVLLFYSTIIAVYFNPLSSHSAEKDTMATVLYTVVTPMLNPFIYSLRNRYLKGA
LKKVV


>gnl|BL_ORD_ID|103003 ENSP00000386138.2 pep
chromosome:GRCh38:1:247757374:247758680:-1 gene:ENSG00000221888.4 transcript:ENST00000408896.4
gene_biotype:protein_coding transcript_biotype:protein_coding gene_symbol:OR1C1
description:olfactory receptor family 1 subfamily C member 1 [Source:HGNC Symbol;Acc:HGNC:8182]

MEKRNLTVVREFVLLGLPSSAEQQHLLSVLFLCMYLATTLGNMLIIATIGFDSHLHSPMY
FFLSNLAFVDICFTSTTVPQMVVNILTGTKTISFAGCLTQLFFFVSFVNMDSLLLCVMAY
DRYVAICHPLHYTARMNLCLCVQLVAGLWLVTYLHALLHTVLIAQLSFCASNIIHHFFCD
LNPLLQLSCSDVSFNVMIIFAVGGLLALTPLVCILVSYGLIFSTVLKITSTQGKQRAVST
CSCHLSVVVLFYGTAIAVYFSPSSPHMPESDTLSTIMYSMVAPMLNPFIYTLRNRDMKRG
LQKMLLKCTVFQQQ


>gnl|BL_ORD_ID|103002 ENSP00000493457.1 pep
chromosome:GRCh38:1:247754846:247760556:-1 gene:ENSG00000221888.4 transcript:ENST00000641256.1
gene_biotype:protein_coding transcript_biotype:protein_coding gene_symbol:OR1C1
description:olfactory receptor family 1 subfamily C member 1 [Source:HGNC Symbol;Acc:HGNC:8182]

MEKRNLTVVREFVLLGLPSSAEQQHLLSVLFLCMYLATTLGNMLIIATIGFDSHLHSPMY
FFLSNLAFVDICFTSTTVPQMVVNILTGTKTISFAGCLTQLFFFVSFVNMDSLLLCVMAY
DRYVAICHPLHYTARMNLCLCVQLVAGLWLVTYLHALLHTVLIAQLSFCASNIIHHFFCD
LNPLLQLSCSDVSFNVMIIFAVGGLLALTPLVCILVSYGLIFSTVLKITSTQGKQRAVST
CSCHLSVVVLFYGTAIAVYFSPSSPHMPESDTLSTIMYSMVAPMLNPFIYTLRNRDMKRG
LQKMLLKCTVFQQQ


>gnl|BL_ORD_ID|106110 ENSP00000305424.2 pep chromosome:GRCh38:16:3188204:3206556:1
gene:ENSG00000168124.3 transcript:ENST00000304646.3 gene_biotype:protein_coding
transcript_biotype:protein_coding gene_symbol:OR1F1
description:olfactory receptor family 1 subfamily F member 1 [Source:HGNC Symbol;Acc:HGNC:8194]

MSGTNQSSVSEFLLLGLSRQPQQQHLLFVFFLSMYLATVLGNLLIILSVSIDSCLHTPMY
FFLSNLSFVDICFSFTTVPKMLANHILETQTISFCGCLTQMYFVFMFVDMDNFLLAVMAY
DHFVAVCHPLHYTAKMTHQLCALLVAGLWVVANLNVLLHTLLMAPLSFCADNAITHFFCD
VTPLLKLSCSDTHLNEVIILSEGALVMITPFLCILASYMHITCTVLKVPSTKGRWKAFST
CGSHLAVVLLFYSTIIAVYFNPLSSHSAEKDTMATVLYTVVTPMLNPFIYSLRNRYLKGA
LKKVV
```

As the above output only contained six sequences, I decided to check my output on the NCBI BLASTp online database. The filters on the online webpage are definitely more limited, so this will impact the output. Below we can see the filter options available Whilst I was able to filter for E values less than 1E-30, I was unable to impose a filter for percentage similarity. Percentage similarity is

distinctly different from query coverage, which measures how much of the query sequence is covered by the alignment. I was tempted to use percentage identity as an approximation for percentage similarity, however according to Hebert et.al (2003) and Desalle et.al (2005) BLAST percent identity scores do not necessarily equal the percent sequence similarity, especially as the sequence get more divergent and become harder to align. With the knowledge of this, I tested how the output changed when limiting percentage identity to greater than 60%, and it provided me with seven outputs – leading me to believe that the errors in my code may have been between an inability to discern between percentage similarity and percentage identity when applying my filters. In light of this, I decided to filter my outputs by E value only.

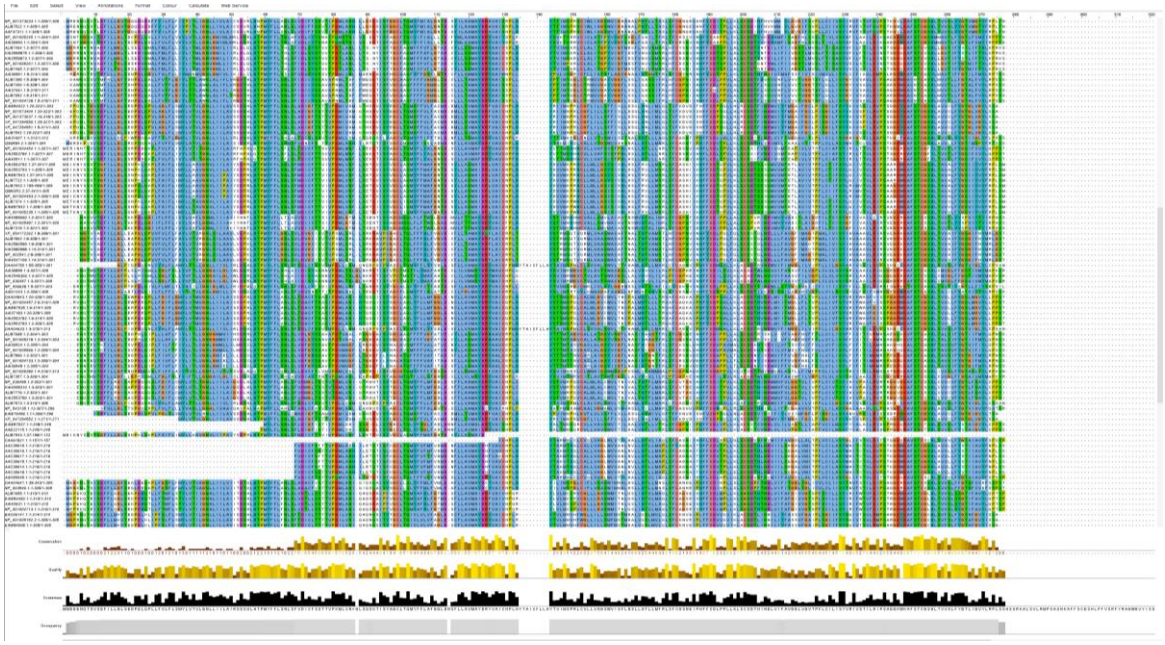


I then selected 250 outputs that matched my criteria, downloaded them into a text file direct from the NCBI BLAST+ webpage and converted the text file into a fasta format for MSA analysis. See below for my code:

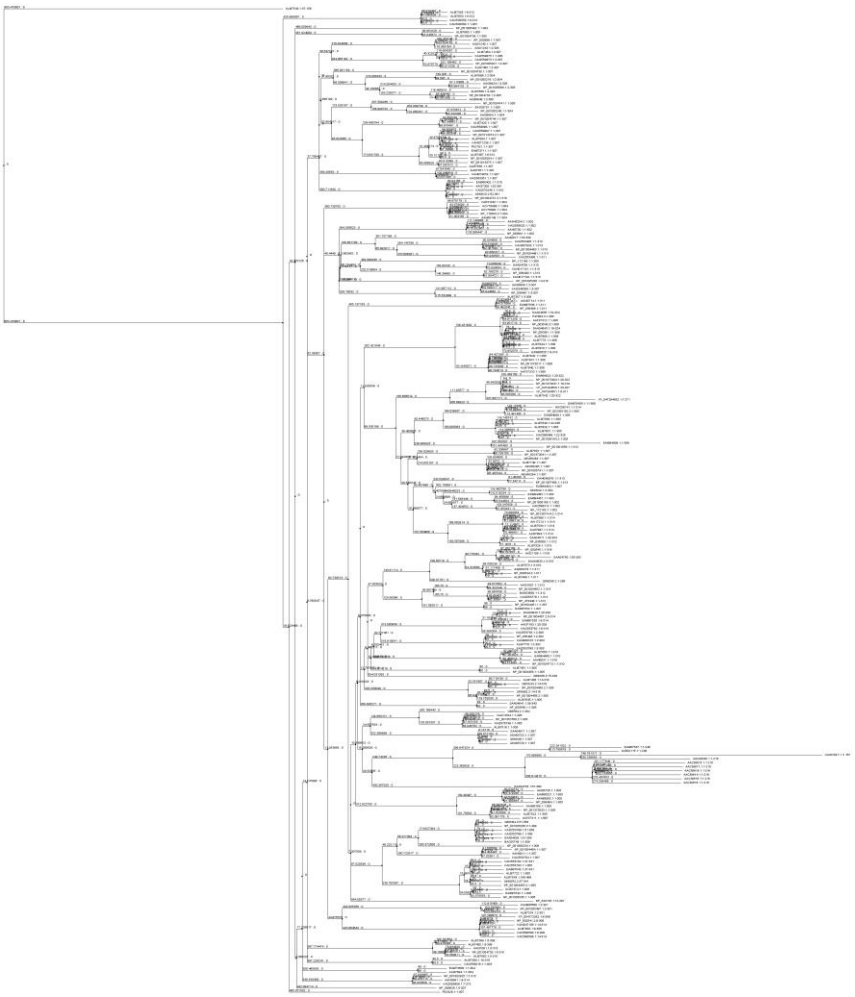
```
#Converting text files to fasta files
def txt_to_fasta(txt_file, fasta_file):
    with open(txt_file, 'r') as f:
        sequence = f.read().replace('\n', '')

    with open(fasta_file, 'w') as f:
        f.write(f">\n{sequence}")

txt_file = "seqdump.txt"
fasta_file = "jalview.fasta"
txt_to_fasta(txt_file, fasta_file)
```

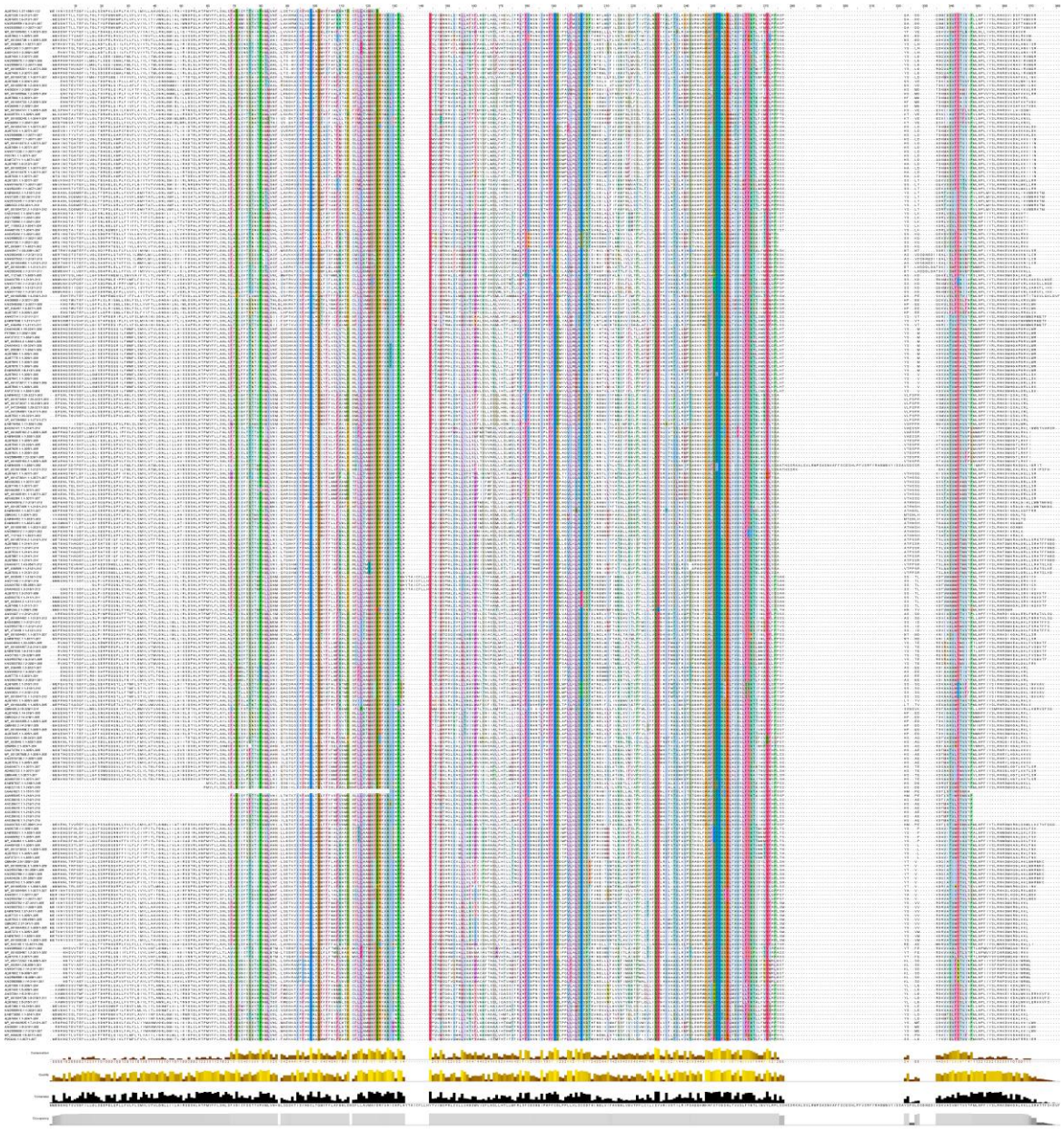
MSA Analysis:

The sequences were analysed and the format edited when needed in jalview to ensure that the fasta file output could be read by the hmmbuild function.



Full versions:





After proofing and editing through the file in Jalview, I then downloaded my MSA fasta data to build the HMM model (and called it jalview.fa).

The Hmm Model:

Expected outputs:

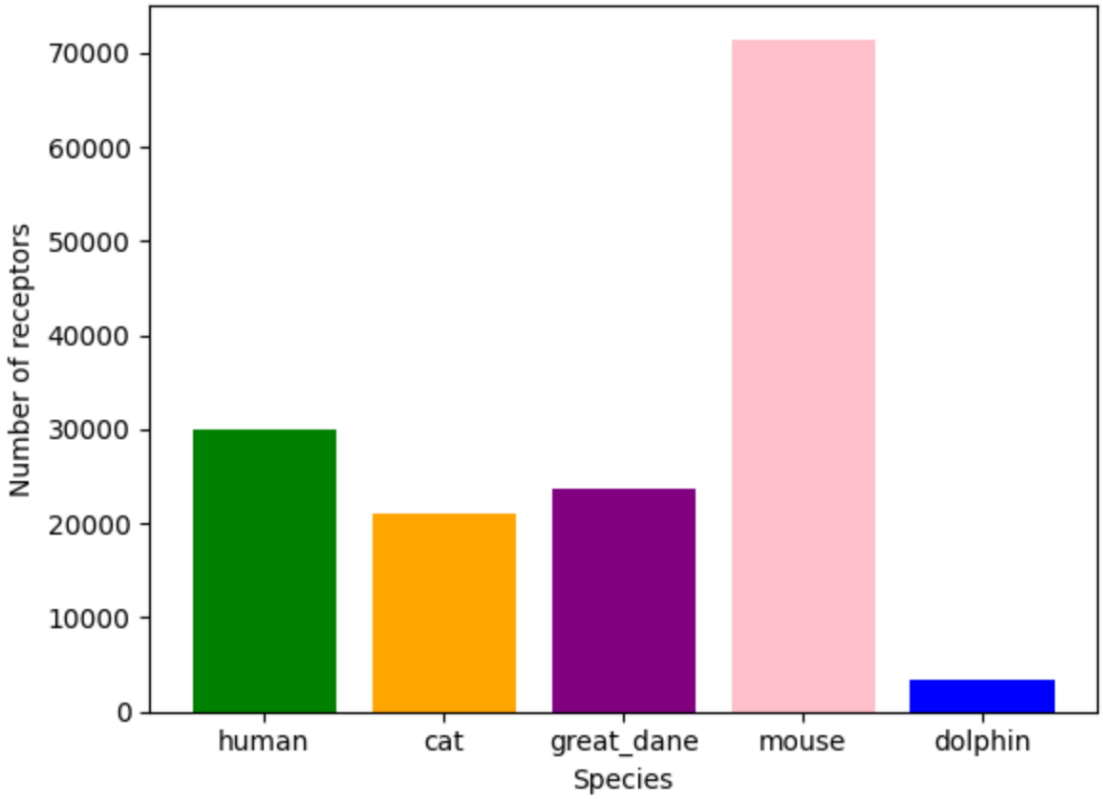
The values listed below are approximate, dependent on the literature cited. However, when reading through publications there appeared to be very little extreme deviations in number of olfactory receptors of the chosen mammals (although dolphin was quite hard to find references to).

- According to Nimura and Nei (2003) humans have almost 400 functional olfactory receptors (Nimura and Nie, 2003).
- According to Young et.al (2003), mice should have between 1200 and 1500 olfactory receptors (Young et.al, 2003).

- According to Quignon et.al (2003) dogs should have around 1070 olfactory receetors, although it in important to note that the average number of olfactory receptors for specific dog breeds can differ. (Quignon et.al, 2003).
- Many reports recorded dolphin olfactory receptor numbers as NaN, however Kremers et.al (2016) reported the value to be around 37, which felt sufficiently low enoigh to accept as an expected value. (Kremers et.al, 2016). Whilst this value might feel low for an aquatic mammal, dolphins rely more heavity on their other senses and also process smells in a completely different way to terrestrial mammals.
- In the literature, exact values for numbers of cat olfactory receptors are no given, instead the number is said to be high.

```
import matplotlib.pyplot as plt

# building HMM
!hmmbuild human.hmm jalview.fa

# cat, dog (great-dane), mouse, dolphin, human

#cat
!hmmsearch human.hmm cat.pep.all.fa > cat.out
#great dane
!hmmsearch human.hmm great_dane.pep.all.fa > great_dane.out
# mouse
!hmmsearch human.hmm Mouse.pep.all.fa > mouse.out
#dolphin
!hmmsearch human.hmm dolphin.pep.all.fa > dolphin.out
#human
!hmmsearch human.hmm human.pep.all.fa > human.out

#plotting
# using parse() filter
def parse(file):
    with open(file, 'r') as f:
        lines = f.read().split('\n')  # split by new lines
        data = [line.split(',') for line in lines if line]  # split by commas and filter out empty lines
        data = [[value for value in row if value] for row in data]  # filter out empty strings in each row
    return len(data)  # return the number of matches

species = ['human','cat', 'great_dane', 'mouse', 'dolphin']
counts = [parse(f"{s}.out") for s in species]

colors = ['green','orange', 'purple', 'pink', 'blue']

plt.bar(species, counts, color = colors)
plt.xlabel('Species')
plt.ylabel('Number of receptors')
plt.show()
```

```
# hmmbuild :: profile HMM construction from multiple sequence alignments
# HMMER 3.3.2 (Nov 2020); http://hmmer.org/
# Copyright (C) 2020 Howard Hughes Medical Institute.
# Freely distributed under the BSD open source license.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# input alignment file:             jalview.fa
# output HMM file:                  human.hmm
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# idx name                  nseq  alen  mlen eff_nseq re/pos description
#---- -------------------- ----- ----- ----- -------- ------ -----------
1     jalview                250   381   307     1.23  0.591

# CPU time: 0.20u 0.00s 00:00:00.20 Elapsed: 00:00:00.20
```




```python
def print_receptor_counts(species, counts):
    for s, c in zip(species, counts):
        print(f"The number of olfactory receptors in {s} is {c}.")

print_receptor_counts(species, counts)
```

```
The number of olfactory receptors in human is 30018.
The number of olfactory receptors in cat is 20980.
The number of olfactory receptors in great_dane is 23646.
The number of olfactory receptors in mouse is 71468.
The number of olfactory receptors in dolphin is 3316.
```

The model appears to over-estimate for all species proportionally, implying that perhaps if the parse() took into account sequence similarity and E-value, the overall outputs for all could have decreased. However, as they remain somewhat proportional I will not re-run the hmmer.

References

- Kute et.al, 2022: doi: 10.3389/fgene.2021.796060
- Claverie et.al, 1997: https://doi.org/10.1016/S0097-8485(96)00039-3
- Xie et.al, 2023: https://doi.org/10.1038/s42003-023-05513-7
- Zlotorynski et.al, 2019: https://doi.org/10.1038/s41580-019-0178-3
- Deonier et.al, 2005: . ISBN 978-0-387-98785-9.
- Alan et.al, pages 991-994, 2021: https://doi.org/10.1093/biosci/biab060
- ICGSC, Nature 432, 2004: https://doi.org/10.1038/nature03154
- Desalle et.al, 2005: doi:10.1098/rstb.2005.1722
- Hebert et.al, 2003: doi: 10.1098/rspb.2002.2218
- Nimura and Nie, 2003: https://doi.org/10.1073/pnas.1635157100
- Young et.al, 2003: https://doi.org/10.1186/gb-2003-4-11-r71
- Quignon et.al, 2003: https://doi.org/10.1186/gb-2003-4-12-r80
- Kremers et.al, 2016: https://doi.org/10.3389/fevo.2016.00049