HelixIT: A comprehensive DNA analysis toolkit

A project report

presented to

Aarohi Chopra

Department of Computer Science,

San José State University

In partial fulfilment

of the requirements for the

course CS 22B

By:

Shwethal Sayeeram Trikannad and Rucha Deo

May 2024

ABSTRACT

Bioinformatics is the intersection of computational and biological sciences. A major subset of bioinformatics is genomics involving deoxyribonucleic acid (DNA) sequencing and analysis. Parallel to the advancements in sequencing technology, analysis tools have also been created. DNA analysis is achieved through object-oriented programming (OOP), which imitates the complexity and adaptability of human cognitive processes. Here, we introduce an OOP based DNA analysis toolbox that will improve and streamline many elements of DNA sequence analysis, making it easier to analyze data comprehensively.

From simple sequence manipulation using kmers, complement and reverse complement, gc percentages and transcripts to complex functions like identifying primers, detecting hairpins, palindrome discovery, open reading frames , translation across all six frames, visualizations like open reading frame visualizer and kmer plots among others; our toolset has a vast range of capabilities. The program also has the added functionality of easing user input by a script that automates sequence accession from Entrez databases using just the accession id. Our project intends to enable scientists and medical professionals to investigate, decipher, and apply the enormous amount of genetic data for biological research and therapeutic uses by offering an extensive range of instruments for DNA analysis.

Keywords: DNA analysis toolkit, object-oriented program, DNA sequencing

Table of contents

I. BACKGROUND

DNA analysis is key in many areas of study. It is at the crux of species identification, gene therapy, identifying inheritable diseases, tracing evolutionary relationships and forensic studies among others. As sequencing technology continues to evolve at a rapid pace it is paramount for analysis tools to keep up ensuring researchers and industry leaders have the necessary arsenal to analyze and interpret data.

One of the first attempts at creating a genomic investigation toolkit was by Sayyab et. al. in 2009 when they created a package specific to the Begomovirus genome [2]. This tool executed DNA, ribonucleic acid (RNA), and protein analysis. Bioconductor [3] in 2017 decided to host a suite of their genome analysis tools on the cloud. This was one of the first instances of a complete toolbox of computational methods housed in a single package accessible on the cloud. The packages are all written in R and can be installed in RStudio. Iqbal et. al. [4] created Bioinformatics Mini Toolbox (BMT) in 2020, a composite of seven different tools for DNA and protein analysis. Steps like sequence trimming, alignment, translation, and protein study can be carried out with this package. DNA methylation is an essential epigenetic function that influences gene expression. A suite of tools named MSuite2 [5] were designed in 2022 to review and visualize this phenomenon in their custom dataset.

Despite the above applications there still exists a need to create a computational toolbox that does an in-depth DNA analysis for any living organism. Our project presents HelixIT, a comprehensive DNA analysis toolkit as the solution. This project is written in Python and utilizes a number of libraries to provide a case-by-case report of each input sequence.

II. CODE DESIGN

The python script is divided into classes with specific attributes and methods under each. There are a total of four classes with DNASequence being the parent class and the other three either its children or associated with it. This parent class is inherited by ComplementarySequence and Translation classes. It is also associated with the EntrezData class as it returns DNASequence in its fetch_sequence method. A total of twenty methods have been created with the vast majority belonging to the parent class. A unified modelling language (UML) diagram showing the code design is seen in Fig. 1.

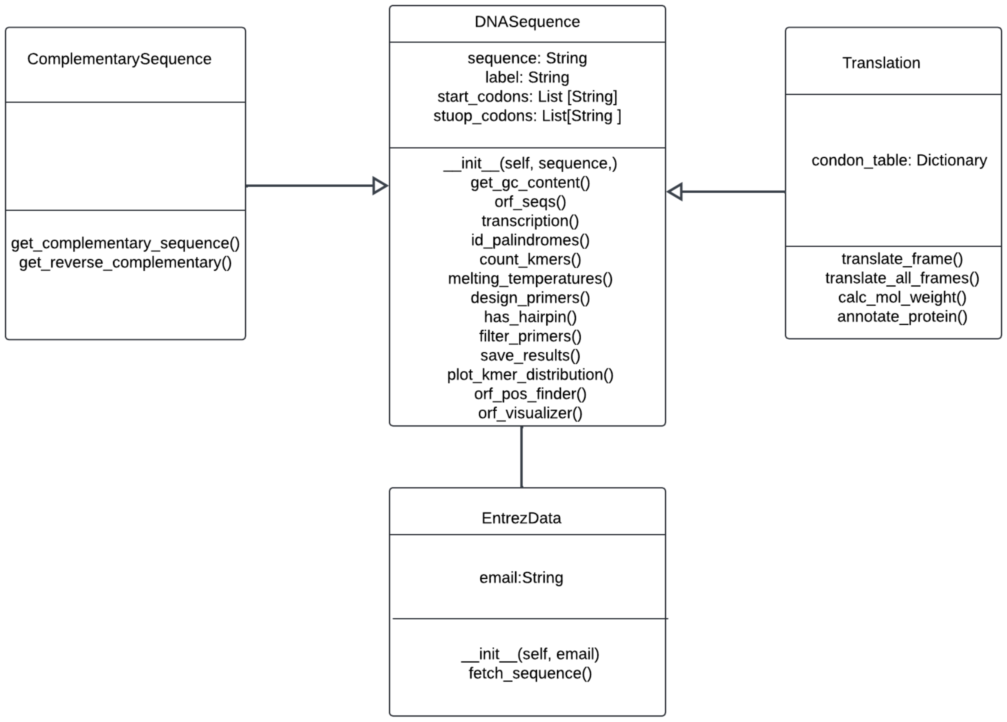
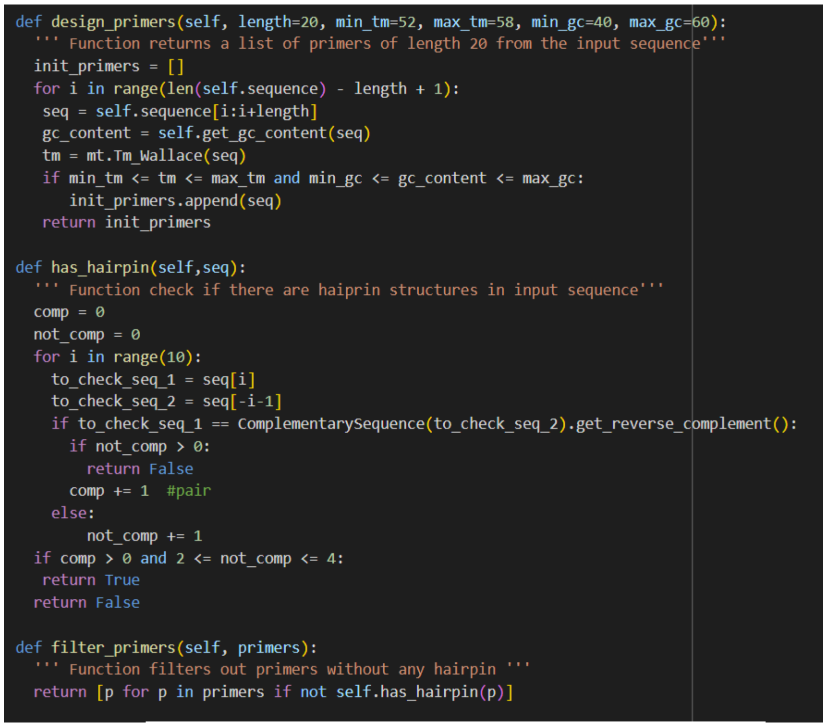


Fig 1. UML of the entire program design

III. Code implementation

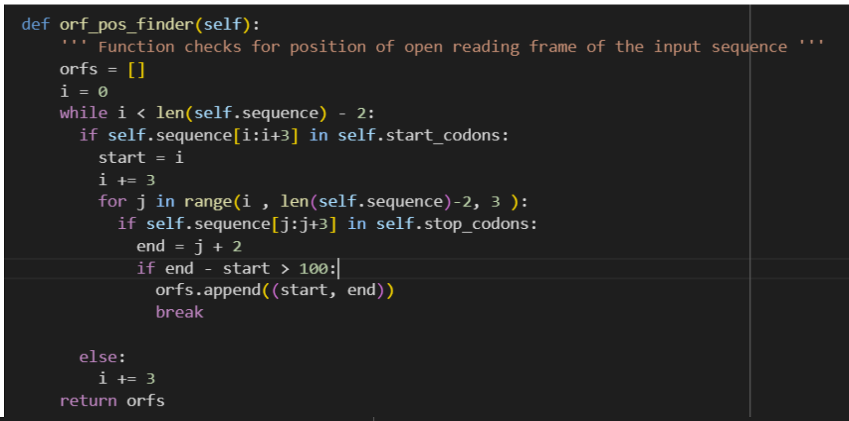
The user is prompted to input their choice of accession ids and the fetch_sequence method in EntrezData class fetches the respective Fasta sequence associated with it. This sequence is then investigated, and results are generated respective to the number of accession ids the user has prompted the script to search. A snippet of a code showing three primer related methods is



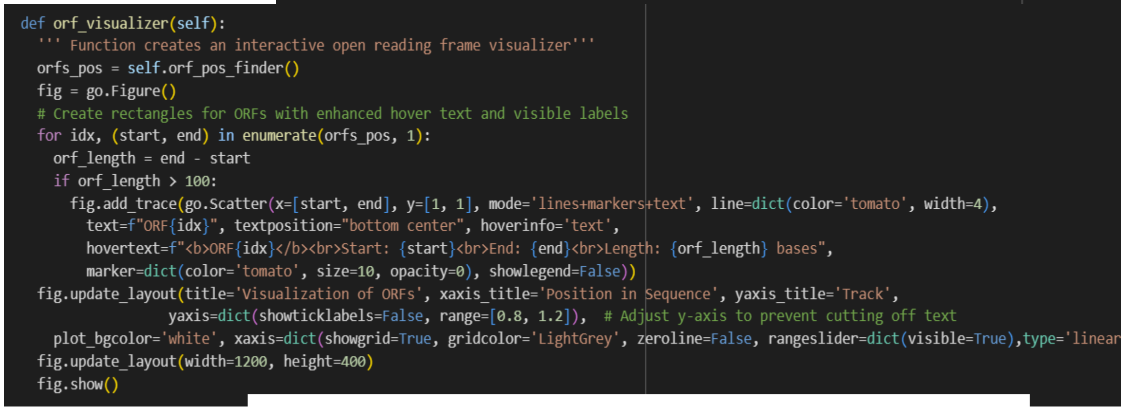
```
def design_primers(self, length=20, min_tm=52, max_tm=58, min_gc=40, max_gc=60):
  ''' Function returns a list of primers of length 20 from the input sequence'''
  init_primers = []
  for i in range(len(self.sequence) - length + 1):
   seq = self.sequence[i:i+length]
   gc_content = self.get_gc_content(seq)
   tm = mt.Tm_Wallace(seq)
   if min_tm <= tm <= max_tm and min_gc <= gc_content <= max_gc:
      init_primers.append(seq)
   return init_primers

def has_hairpin(self,seq):
  ''' Function check if there are haiprin structures in input sequence'''
  comp = 0
  not_comp = 0
  for i in range(10):
    to_check_seq_1 = seq[i]
    to_check_seq_2 = seq[-i-1]
    if to_check_seq_1 == ComplementarySequence(to_check_seq_2).get_reverse_complement():
      if not_comp > 0:
        return False
      comp += 1  #pair
    else:
        not_comp += 1
  if comp > 0 and 2 <= not_comp <= 4:
   return True
  return False

def filter_primers(self, primers):
  ''' Function filters out primers without any hairpin '''
  return [p for p in primers if not self.has_hairpin(p)]
```

Fig. 2 Primer methods

displayed in Fig. 2. Primers are short nucleotide sequences added to genetic material during library preparation and get accidentally sequenced sometimes. This list of functions checks for and returns any primers present in the input DNA sequence. The first function design_primers uses default length, maximum and minimum temperatures and maximum and minimum gc_percentages to find initial primer candidates in the input sequence. After initializing an empty list named init_primers, the function then iterates through a range spanning from zero to length of sequence minus the primer length plus one to account for desired length of primer and stop exclusivity. A Biopython [5] method named TM_Wallace under melting temperature module is used to calculate the temperature of the primers. The function returns a list of initial primer candidates which become the input for the filter_primers function. Hairpins [6] [7] are formed when the ends of a single stranded DNA sequence are complementary to each other leaving behind 4-8 non-complementary free bases in the middle. These are undesirable structures that creep up during primer formation and DNA , RNA sequencing. The has_hairpin function returns a Boolean with True corresponding to presence of hairpin and False signifying its absence. This function utilizes a has_hairpin function to check for any hairpin structures in the primer candidates. It executes simple list comprehension to do so. Fig 3. highlights another important function used to identify and visualize open reading frames in the sequence. This function initiates a while and for loop to discover start and stop codon positions and includes only open reading frames greater than 100 bases [8] as is followed by



```
def orf_pos_finder(self):
  ''' Function checks for position of open reading frame of the input sequence '''
  orfs = []
  i = 0
  while i < len(self.sequence) - 2:
    if self.sequence[i:i+3] in self.start_codons:
      start = i
      i += 3
      for j in range(i , len(self.sequence)-2, 3 ):
        if self.sequence[j:j+3] in self.stop_codons:
          end = j + 2
          if end - start > 100:
            orfs.append((start, end))
            break

    else:
      i += 3
  return orfs
```



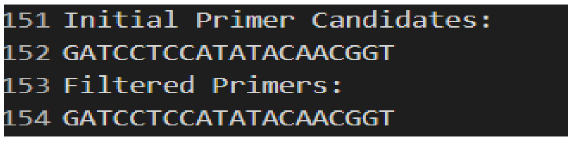
```
def orf_visualizer(self):
  ''' Function creates an interactive open reading frame visualizer'''
  orfs_pos = self.orf_pos_finder()
  fig = go.Figure()
  # Create rectangles for ORFs with enhanced hover text and visible labels
  for idx, (start, end) in enumerate(orfs_pos, 1):
    orf_length = end - start
    if orf_length > 100:
      fig.add_trace(go.Scatter(x=[start, end], y=[1, 1], mode='lines+markers+text', line=dict(color='tomato', width=4),
        text=f"ORF{idx}", textposition="bottom center", hoverinfo='text',
        hovertext=f"<b>ORF{idx}</b><br>Start: {start}<br>End: {end}<br>Length: {orf_length} bases",
        marker=dict(color='tomato', size=10, opacity=0), showlegend=False))
  fig.update_layout(title='Visualization of ORFs', xaxis_title='Position in Sequence', yaxis_title='Track',
                    yaxis=dict(showticklabels=False, range=[0.8, 1.2]),  # Adjust y-axis to prevent cutting off text
    plot_bgcolor='white', xaxis=dict(showgrid=True, gridcolor='LightGrey', zeroline=False, rangeslider=dict(visible=True),type='linear
  fig.update_layout(width=1200, height=400)
  fig.show()
```

Fig 3. Open Reading Frame methods

popular sequencing platforms like Illumina. The next function orf_visualizer implements the above function and the library Plotly to map out an interactive and detailed open reading frame plot. Instructions to run the entire program are included in the appendix.

IV. Results

Upon running the program, a function named save_results opens a file which can be accessed on the local google colab environment and writes most results to it. The two visualizations are returned to the screen. Fig 4. is the output obtained after running the primer functions on accession id U49845 [9]. In this case, as the initial primer candidate does not have a hairpin it is not filtered out and is returned as is. Fig 5a. is a bar plot showing the kmer percentage across the sequence and Fig 5b. is an interactive open reading frame visualizer which is the output of the code run in Fig 3. The barplot, created with Seaborn, depicts the distribution of kmers across the entire sequence. The X-axis shows all the possible kmers of length three, that can be procured, and the Y-axis is the frequency of respective kmers. The most popular kmer is TTT with a frequency greater than 200 and the least is GGG with frequency less than 25 occurrences. The plot in Fig 5b. is an interactive plot with tracks signifying the open reading frames. The X-axis has numbers spanning from 0 to 5028 encompassing the entire sequence length. The orange-colored track



```
Initial Primer Candidates:
GATCCTCCATATACAACGGT
Filtered Primers:
GATCCTCCATATACAACGGT
```

Fig 4. Output of methods run in Fig 2.

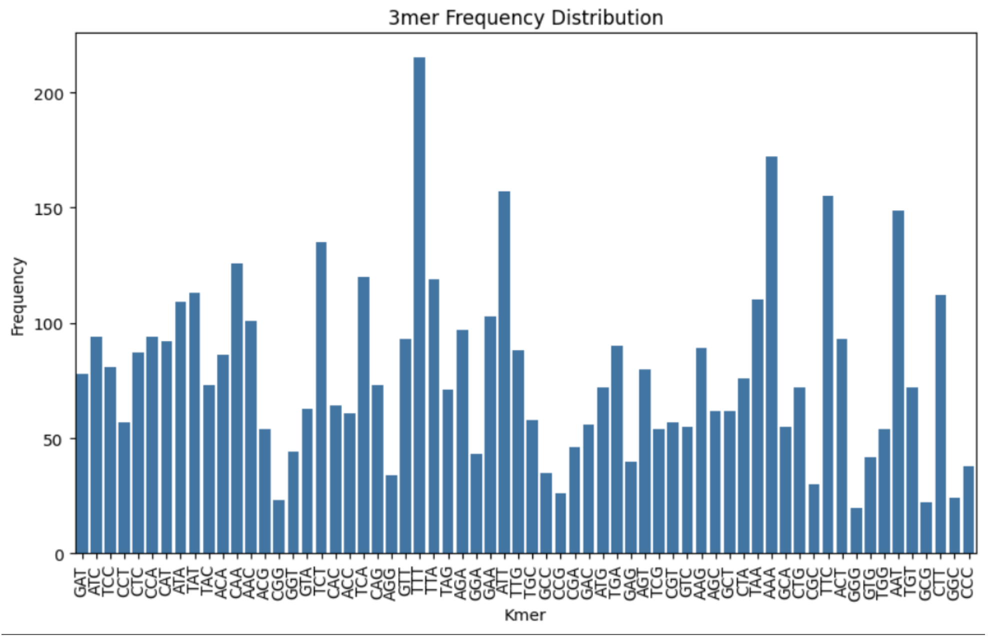


Fig 5a. Barplot showing Kmer distribution.

signifies exons, and the breaks introns (non-coding areas). A sliding bar below the X-axis helps users

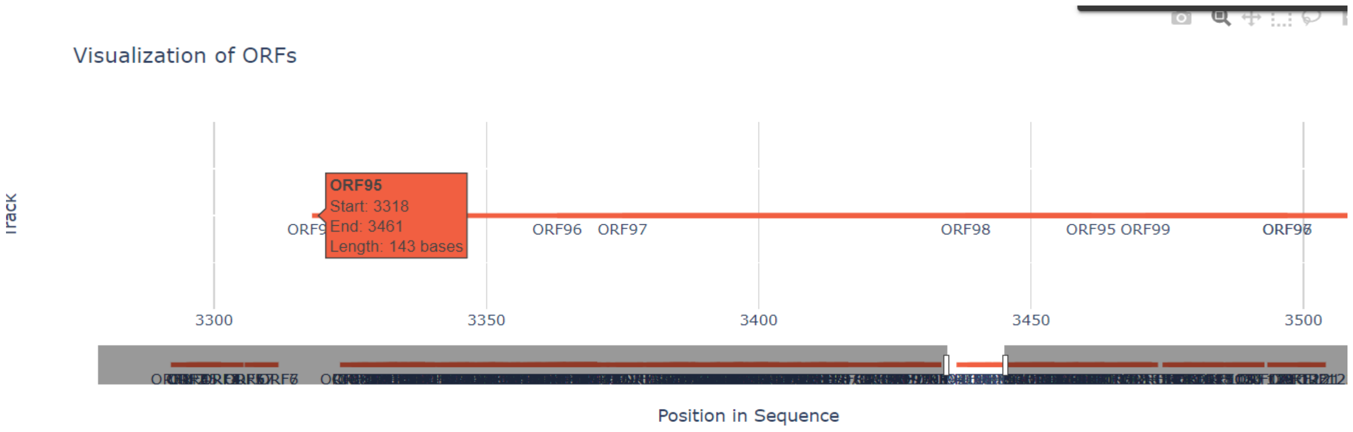


Fig 5b. Open Reading Frame Visualizer (output of methods run in Fig 3)

isolate a specific region to acquire a more detailed picture of the reading frame instance. Details like positions of start and stop codons along with the length of the frame are visible when the pointer is placed on the track.

V. Conclusion

Our DNA analysis system provides comprehensive features such as ORF identification and visualization, primer detection and filtering on the basis of hairpins, translation across all six frames and palindrome recognition among several other functions. Future enhancements include integration of publicly available application programming interface (APIs) like InterPro to enhance protein analysis and execute machine learning for predictive modeling. Hosting the system on an open-source platform like Streamlit will improve accessibility and user experience. These advancements aim to thrust genetic research forward and aid in precision medicine and gene therapy.

References

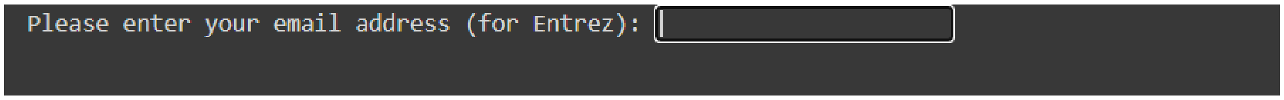
1. Luscombe, Nicholas M., Dov Greenbaum, and Mark Gerstein. 2001. "What is bioinformatics? An introduction and overview." Yearbook of medical informatics 10.01 (February 2001), 83-100.
2. S. Sayyab, A. Nadeem and S. Fazal. 2009. "BVirusGS: DNA analysis toolkit for Begomovirus genome analysis," 2009 IEEE 13th International Multitopic Conference, Islamabad, Pakistan, (December 2009) 1-5, doi: 10.1109/INMIC.2009.5383141.
3. Gibbs, W. Wayt. "A test drive of a DNA-analysis toolkit in the cloud." Nature 552.7683 (2017): 137-139.
4. Muhammad Nasir Iqbal, Muhammad Asif Rasheed, Muhammad Awais, Wathek Chammam, Sumaira Kanwal, Sami Ullah Khan, Salina Saddick, Iskander Tlili, BMT: Bioinformatics mini toolbox for comprehensive DNA and protein analysis,Genomics, Volume 112, Issue 6, 2020, Pages 4561-4566, ISSN 0888-7543, https://doi.org/10.1016/j.ygeno.2020.08.010
5. Cock PJ, Antao T, Chang JT, et al. 2009. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. (June 2009), 1422-1423. doi:10.1093/bioinformatics/btp163
6. Keiichi Ohshima, Robert D. Wells, Hairpin Formation during DNA Synthesis Primer Realignment in Vitro in Triplet Repeat Sequences from Human Hereditary Disease Genes*,Journal of Biological Chemistry, Volume 272, Issue 27,1997, 16798-16806, ISSN 0021-9258, https://doi.org/10.1074/jbc.272.27.16798
7. Zhang, Y. (2013). Hairpin Structure. In: Dubitzky, W., Wolkenhauer, O., Cho, KH., Yokota, H. (eds) Encyclopedia of Systems Biology. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-9863-7_325
8. Woodcroft BJ, Boyd JA, Tyson GW. 2016. OrfM: a fast open reading frame predictor for metagenomic data. Bioinformatics. (September 2016), 2702-2703. doi:10.1093/bioinformatics/btw241

9. L.E. Torpey, P.E. Gibbs, J. Nelson, and C.W. Lawrence. 1999. Sample genbank record. (June 1999). Retrieved May 12, 2024 from https://www.ncbi.nlm.nih.gov/genbank/samplerecord/

10. Primer design : https://www.premierbiosoft.com/tech_notes/PCR_Primer_Design.html

APPENDIX

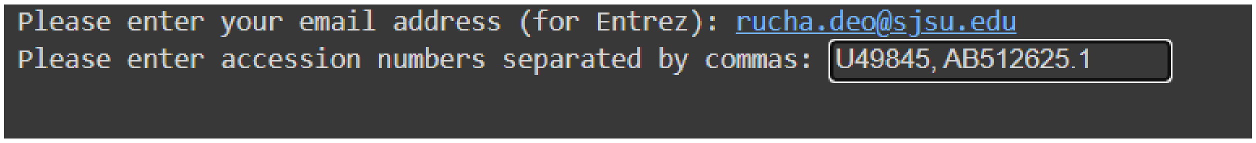
Instructions to run code: Upon running the code cell containing the main function, the user is prompted to input an email id previously used to access Entrez database. Fig 6. shows a representation of this.



```
Please enter your email address (for Entrez):
```

Fig 6. Email input prompt.

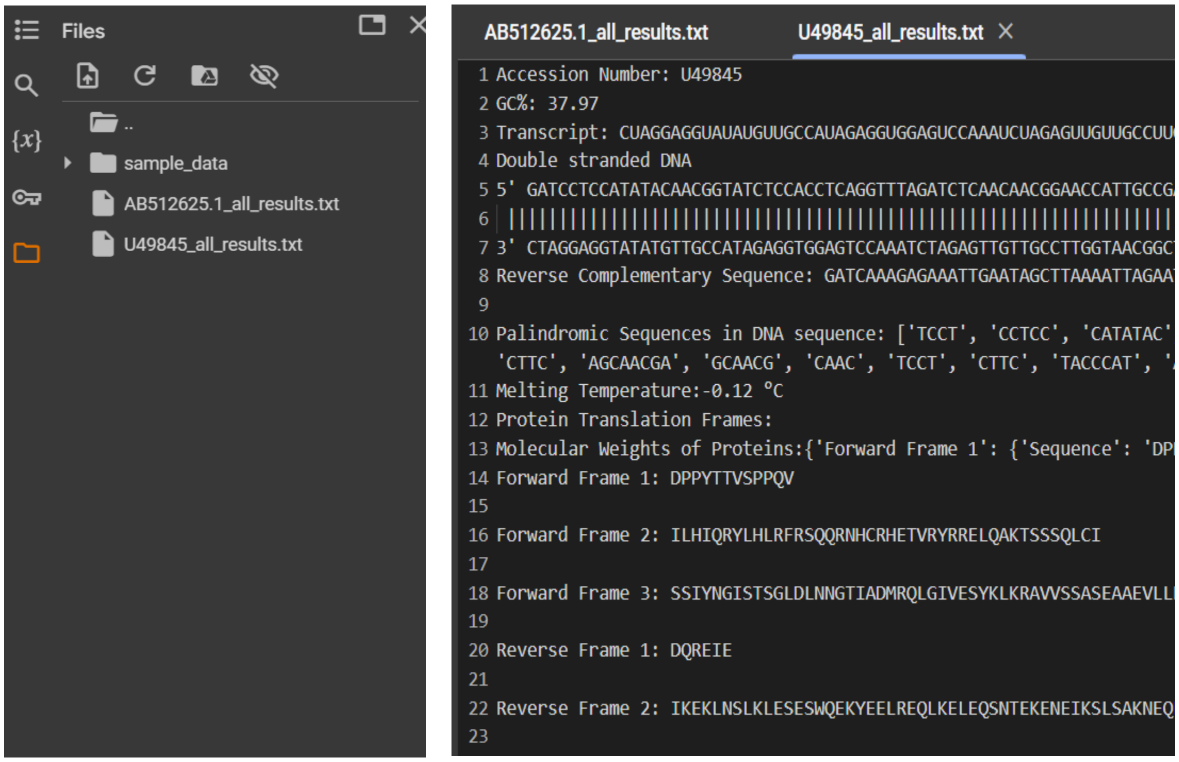
The user is then prompted to input accession numbers separated by a comma. Fig 7. shows a sample of this.



```
Please enter your email address (for Entrez): rucha.deo@sjsu.edu
Please enter accession numbers separated by commas: U49845, AB512625.1
```

Fig 7. Accession id input prompt.

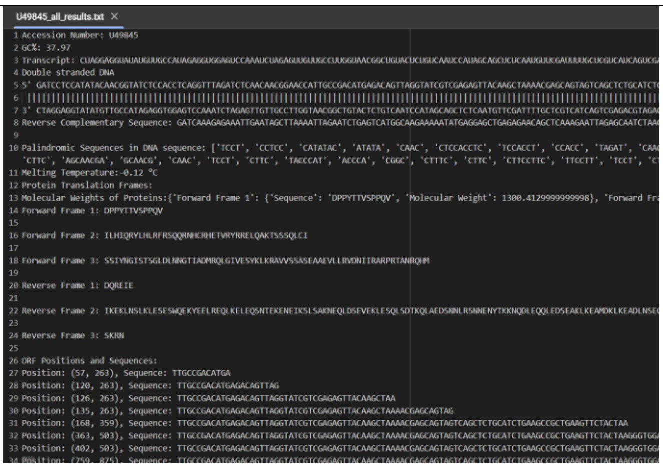
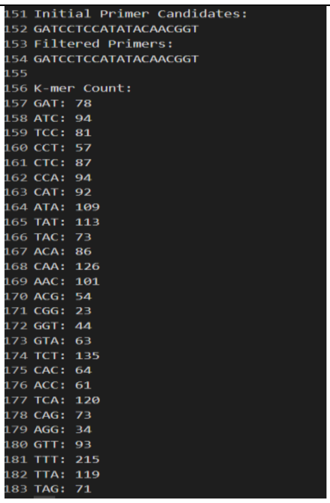
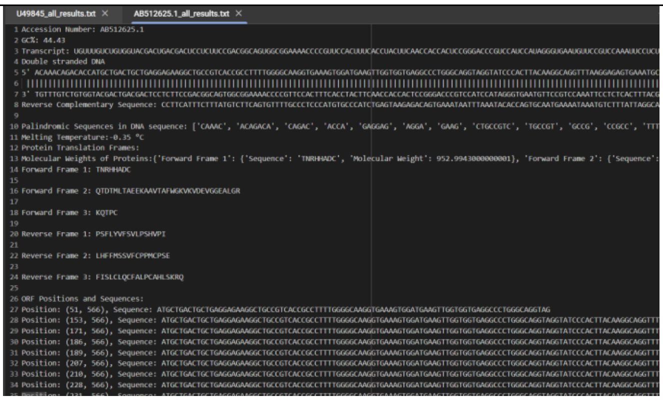
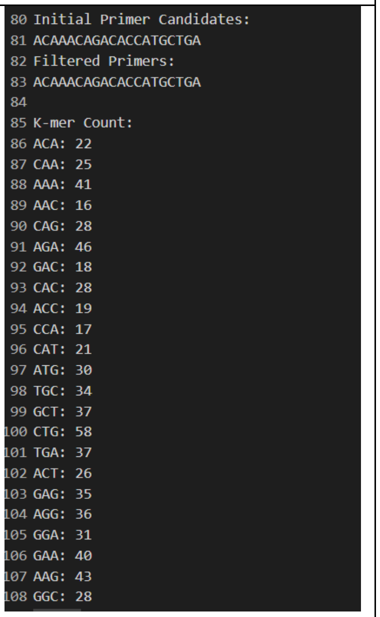
The code then runs and outputs both visualizations to the screen. The text file for each accession id can be obtained on the left-hand side of the screen. Fig 8. depicts this.



```
Accession Number: U49845
GC%: 37.97
Transcript: CUAGGAGGUAUAUGUUGCCAUAGAGGUGGAGUCCAAAUCUAGAGUUGUUGCCUU
Double stranded DNA
5' GATCCTCCATATACAACGGTATCTCCACCTCAGGTTTAGATCTCAACAACGGAACCATTGCCG
 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
3' CTAGGAGGTATATGTTGCCATAGAGGTGGAGTCCAAATCTAGAGTTGTTGCCTTGGTAACGGC
Reverse Complementary Sequence: GATCAAAGAGAAATTGAATAGCTTAAAATTAGAA

Palindromic Sequences in DNA sequence: ['TCCT', 'CCTCC', 'CATATAC'
 'CTTC', 'AGCAACGA', 'GCAACG', 'CAAC', 'TCCT', 'CTTC', 'TACCCAT', '
Melting Temperature:-0.12 °C
Protein Translation Frames:
Molecular Weights of Proteins:{'Forward Frame 1': {'Sequence': 'DP
Forward Frame 1: DPPYTTVSPPQV

Forward Frame 2: ILHIQRYLHLRFRSQQRNHCRHETVRYRRELQAKTSSSQLCI

Forward Frame 3: SSIYNGISTSGLDLNNGTIADMRQLGIVESYKLKRAVVSSASEAAEVLL

Reverse Frame 1: DQREIE

Reverse Frame 2: IKEKLNSLKLESESWQEKYEELREQLKELEQSNTEKENEIKSLSAKNEQ

```

Fig 8. Output file access and results.

Table of results: Table 1. represents a snippet of outputs secured when two accession ids U49845 belonging to saccharomyces cerevisiae and AB512625.1 which is the id for beta hemoglobin gene for cattle. The complete report and graphs can be accessed in the python notebook submitted with this project upon running the two accession ids.

Accession Id	Results part 1	Results part 2
U49845		
AB512625.1		

```
U49845_all_results.txt ×
Accession Number: U49845
GC%: 37.97
Transcript: CUAGGAGGUAUAUGUUGCCAUAGAGGUGGAGUCCAAAUCUAGAGUUGUUGCCUUGGUAACGGCUGUACUCUGUCAAUCCAUAGCAGCUCUCAAUGUUCGAUUUUGCUCGUCAUCAGUCGA
Double stranded DNA
5' GATCCTCCATATACAACGGTATCTCCACCTCAGGTTTAGATCTCAACAACGGAACCATTGCCGACATGAGACAGTTAGGTATCGTCGAGAGTTACAAGCTAAAACGAGCAGTAGTCAGCTCTGCATCTG
 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
3' CTAGGAGGTATATGTTGCCATAGAGGTGGAGTCCAAATCTAGAGTTGTTGCCTTGGTAACGGCTGTACTCTGTCAATCCATAGCAGCTCTCAATGTTCGATTTTGCTCGTCATCAGTCGAGACGTAGAC
Reverse Complementary Sequence: GATCAAAGAGAAATTGAATAGCTTAAAATTAGAATCTGAGTCATGGCAAGAAAAATATGAGGAGCTGAGAGAACAGCTCAAAGAATTAGAGCAATCTAAC

Palindromic Sequences in DNA sequence: ['TCCT', 'CCTCC', 'CATATAC', 'ATATA', 'CAAC', 'CTCCACCTC', 'TCCACCT', 'CCACC', 'TAGAT', 'CAAC
 'CTTC', 'AGCAACGA', 'GCAACG', 'CAAC', 'TCCT', 'CTTC', 'TACCCAT', 'ACCCA', 'CGGC', 'CTTTC', 'CTTC', 'CTTCCTTC', 'TTCCTT', 'TCCT', 'C
Melting Temperature:-0.12 °C
Protein Translation Frames:
Molecular Weights of Proteins:{'Forward Frame 1': {'Sequence': 'DPPYTTVSPPQV', 'Molecular Weight': 1300.4129999999998}, 'Forward Fra
Forward Frame 1: DPPYTTVSPPQV

Forward Frame 2: ILHIQRYLHLRFRSQQRNHCRHETVRYRRELQAKTSSSQLCI

Forward Frame 3: SSIYNGISTSGLDLNNGTIADMRQLGIVESYKLKRAVVSSASEAAEVLLRVDNIIRARPRTANRQHM

Reverse Frame 1: DQREIE

Reverse Frame 2: IKEKLNSLKLESESWQEKYEELREQLKELEQSNTEKENEIKSLSAKNEQLDSEVEKLESQLSDTKQLAEDSNNLRSNNENYTKKNQDLEQQLEDSEAKLKEAMDKLKEADLNSE

Reverse Frame 3: SKRN

ORF Positions and Sequences:
Position: (57, 263), Sequence: TTGCCGACATGA
Position: (120, 263), Sequence: TTGCCGACATGAGACAGTTAG
Position: (126, 263), Sequence: TTGCCGACATGAGACAGTTAGGTATCGTCGAGAGTTACAAGCTAA
Position: (135, 263), Sequence: TTGCCGACATGAGACAGTTAGGTATCGTCGAGAGTTACAAGCTAAAACGAGCAGTAG
Position: (168, 359), Sequence: TTGCCGACATGAGACAGTTAGGTATCGTCGAGAGTTACAAGCTAAAACGAGCAGTAGTCAGCTCTGCATCTGAAGCCGCTGAAGTTCTACTAA
Position: (363, 503), Sequence: TTGCCGACATGAGACAGTTAGGTATCGTCGAGAGTTACAAGCTAAAACGAGCAGTAGTCAGCTCTGCATCTGAAGCCGCTGAAGTTCTACTAAGGGTGGA
Position: (402, 503), Sequence: TTGCCGACATGAGACAGTTAGGTATCGTCGAGAGTTACAAGCTAAAACGAGCAGTAGTCAGCTCTGCATCTGAAGCCGCTGAAGTTCTACTAAGGGTGGA
Position: (759, 875), Sequence: TTGCCGACATGAGACAGTTAGGTATCGTCGAGAGTTACAAGCTAAAACGAGCAGTAGTCAGCTCTGCATCTGAAGCCGCTGAAGTTCTACTAAGGGTGGA
```

```
Initial Primer Candidates:
GATCCTCCATATACAACGGT
Filtered Primers:
GATCCTCCATATACAACGGT

K-mer Count:
GAT: 78
ATC: 94
TCC: 81
CCT: 57
CTC: 87
CCA: 94
CAT: 92
ATA: 109
TAT: 113
TAC: 73
ACA: 86
CAA: 126
AAC: 101
ACG: 54
CGG: 23
GGT: 44
GTA: 63
TCT: 135
CAC: 64
ACC: 61
TCA: 120
CAG: 73
AGG: 34
GTT: 93
TTT: 215
TTA: 119
TAG: 71
```

```
U49845_all_results.txt ×    AB512625.1_all_results.txt ×
Accession Number: AB512625.1
GC%: 44.43
Transcript: UGUUUGUCUGUGGUACGACUGACGACUCCUCUUCCGACGGCAGUGGCGGAAAACCCCGUUCCACUUUCACCUACUUCAACCACCACUCCGGGACCCGUCCAUCCAUAGGGUGAAUGUUCCGUCCAAAUUCCUCU
Double stranded DNA
5' ACAAACAGACACCATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAGGTATCCCACTTACAAGGCAGGTTTAAGGAGAGTGAAATGC
 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
3' TGTTTGTCTGTGGTACGACTGACGACTCCTCTTCCGACGGCAGTGGCGGAAAACCCCGTTCCACTTTCACCTACTTCAACCACCACTCCGGGACCCGTCCATCCATAGGGTGAATGTTCCGTCCAAATTCCTCTCACTTTACG
Reverse Complementary Sequence: CCTTCATTTCTTTATGTCTTCAGTGTTTTGCCCTCCCATGTGCCCATCTGAGTAAGAGACAGTGAAATAATTTAAATACACCAGTGCAATGAAAATAAATGTCTTTATTAGGCA

Palindromic Sequences in DNA sequence: ['CAAAC', 'ACAGACA', 'CAGAC', 'ACCA', 'GAGGAG', 'AGGA', 'GAAG', 'CTGCCGTC', 'TGCCGT', 'GCCG', 'CCGCC', 'TTT
Melting Temperature:-0.35 °C
Protein Translation Frames:
Molecular Weights of Proteins:{'Forward Frame 1': {'Sequence': 'TNRHHADC', 'Molecular Weight': 952.9943000000001}, 'Forward Frame 2': {'Sequence':
Forward Frame 1: TNRHHADC

Forward Frame 2: QTDTMLTAEEKAAVTAFWGKVKVDEVGGEALGR

Forward Frame 3: KQTPC

Reverse Frame 1: PSFLYVFSVLPSHVPI

Reverse Frame 2: LHFFMSSVFCPPMCPSE

Reverse Frame 3: FISLCLQCFALPCAHLSKRQ

ORF Positions and Sequences:
Position: (51, 566), Sequence: ATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAG
Position: (153, 566), Sequence: ATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAGGTATCCCACTTACAAGGCAGGTTT
Position: (171, 566), Sequence: ATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAGGTATCCCACTTACAAGGCAGGTTT
Position: (186, 566), Sequence: ATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAGGTATCCCACTTACAAGGCAGGTTT
Position: (189, 566), Sequence: ATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAGGTATCCCACTTACAAGGCAGGTTT
Position: (207, 566), Sequence: ATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAGGTATCCCACTTACAAGGCAGGTTT
Position: (210, 566), Sequence: ATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAGGTATCCCACTTACAAGGCAGGTTT
Position: (228, 566), Sequence: ATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAGGTATCCCACTTACAAGGCAGGTTT
Position: (231, 566), Sequence: ATGCTGACTGCTGAGGAGAAGGCTGCCGTCACCGCCTTTTGGGGCAAGGTGAAAGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGGTAGGTATCCCACTTACAAGGCAGGTTT
```

```
Initial Primer Candidates:
ACAAACAGACACCATGCTGA
Filtered Primers:
ACAAACAGACACCATGCTGA

K-mer Count:
ACA: 22
CAA: 25
AAA: 41
AAC: 16
CAG: 28
AGA: 46
GAC: 18
CAC: 28
ACC: 19
CCA: 17
CAT: 21
ATG: 30
TGC: 34
GCT: 37
CTG: 58
TGA: 37
ACT: 26
GAG: 35
AGG: 36
GGA: 31
GAA: 40
AAG: 43
GGC: 28
```

Table 1. Table showing snippet of outputs derived from test runs on accession ids U49845 and AB512625.1