2023

BIOCOMPUTING FINAL PROJECT

SUBMITTED BY: ONEZA HASSAN ALVI

REG NUM: 04281913032

BSBIF 8TH

PROJECT: PROTEIN SEQUENCE ANALYSIS USING BIOPYTHON

SUBMITTED TO: Dr. ADNAN AHMAD ANSARI

NCB, QAU ISLAMABAD

5th June 2023

Table of Contents:

Project statement:

Protein Sequence Analysis: Write a Python program that takes a protein sequence as input and performs various analyses on it, such as identifying the amino acid composition, calculating the molecular weight, and predicting the secondary structure.

Analysis and design of program:

This code is a Python script that performs some basic analysis on a protein sequence given in a fasta file by the user. Here is a summary of what the code does under the analysis heading:

- Importing BioPython modules: The code imports three modules from the BioPython library: SeqIO, ProtParam, and ProtParamData. These modules provide functions and data for working with biological sequences and calculating protein properties.

- Asking for user input: The code asks the user for the name of the protein fasta file/ path of the file and stores it in a variable called file_name. It then uses the SeqIO.read function to read the protein sequence from the file and store it as a SeqRecord object in a variable called protein.

- Calculating amino acid composition: The code uses the ProtParam.ProteinAnalysis class to create an object that can calculate various properties of the protein sequence. It then uses the get_amino_acids_percent method to get a dictionary of amino acid composition, where each key is an amino acid and each value is its percentage in the sequence.

- Calculating molecular weight: The program uses another method of ProteinAnalysis, molecular_weight, which returns the molecular weight of the protein in Daltons (Da).

- The program uses a simple algorithm based on the Kyte-Doolittle hydrophobicity scale, which assigns a numerical value to each amino acid based on its hydrophobicity. The program uses a sliding window approach to calculate the mean hydrophobicity for each window of a given size (11 in this case) and compares it to a threshold value (1.6 in this case) to assign helix regions. The program stores the start and end indices of each helix region in a list and prints them at the end. The code uses variables to store the window size (kd_window), the threshold (kd_threshold), the list of helix regions (kd_helix), and the list of hydrophobicity values (kd_value).

- The code prints the results of the analysis, including the protein sequence, the amino acid composition, the molecular weight, and the predicted helix regions using the Kyte-Doolittle scale.

Input:

- The user needs to provide the name of the protein fasta file from the working directory or the whole path to the file. A fasta file is a text file that contains one or more sequences in a specific format.

Output:

The code prints the following information about the protein sequence:

- The protein sequence itself, which is a string of amino acid letters.
- The amino acid composition in the protein sequence, which is a dictionary of amino acid letters and their percentage in the sequence.
- The molecular weight of the protein, which is a numerical value in Daltons (Da).
- The predicted helix regions using Kyte-Doolittle scale, which is a list of tuples of start and end indices of the regions that have a mean hydrophobicity value above a certain threshold. Hydrophobicity is a measure of how water-repellent an amino acid is, and it can be used to predict the secondary structure of proteins.

Built in python functions that are used:

Some of the built-in functions used in this python code are:

- Input(): This function takes a string as an argument and returns the user input as a string.

- Len(): This function takes an iterable object (such as a string, list, tuple, etc.) as an argument and returns the number of elements in it.

- Range(): This function takes one or more integers as arguments and returns an iterable object that represents a sequence of numbers from the start (default 0) to the stop (exclusive) with a given step (default 1).

- Print(): This function takes one or more objects as arguments and writes them to the standard output (such as the screen) separated by a space and followed by a newline. It can also take keyword arguments such as sep, end, file, flush, etc. to modify the output behavior.

Biopython functions used in the program:

Some of the BioPython functions used in this program are:

- SeqIO.read(): This function takes a file name and a format as arguments and returns a SeqRecord object that represents a single sequence from the file. It can also take other arguments such as alphabet, seq_count, etc. to modify the behavior. It is useful for reading files that contain only one sequence, such as fasta files. For files that contain multiple sequences, such as genbank files, SeqIO.parse can be used instead.

- ProtParam.ProteinAnalysis(): This function takes a protein sequence as an argument and returns an instance of the ProteinAnalysis class, which has various methods for calculating different properties of proteins, such as amino acid composition, molecular weight, isoelectric point, secondary structure prediction, etc. It can also take other arguments such as monoisotopic to modify the behavior.

- ProtParamData.kd(): This is a dictionary that contains the Kyte-Doolittle hydrophobicity scale values for each amino acid symbol. It is useful for predicting the secondary structure of proteins based on their hydrophobicity.

Source code:

```
#Oneza Hassan Alvi
#04281913032
#biocomputing project
# Importing Biopython modules

print("ONEZA HASSAN ALVI")
print("REG NUM: 04281913032")
print("BIOCOMPUING FINAL PROJECT")
print("--------------------------------")

from Bio import SeqIO
from Bio.SeqUtils import ProtParam
from Bio.SeqUtils.ProtParamData import kd
```

```
# Asking the user for the protein fasta file name and storing it in a variable
file_name = input("Enter the accurate protein fasta file name from the working
directory or give the whole path: ")

# Reading the protein sequence from the file name given by the user
protein = SeqIO.read(file_name, "fasta") # This function reads a single sequence
from a fasta file and returns a SeqRecord object

# Calculating the amino acid composition
aa_comp = ProtParam.ProteinAnalysis(str(protein.seq)).get_amino_acids_percent()

# Calculating the molecular weight
mw = ProtParam.ProteinAnalysis(str(protein.seq)).molecular_weight()

# Predicting the secondary structure using Kyte-Doolittle hydrophobicity scale
kd_window = 11 # window size for sliding average
kd_threshold = 1.6 # threshold for assigning helix
kd_helix = [] # list of helix regions
kd_value = [] # list of hydrophobicity values

# Calculating the mean hydrophobicity for each window
for i in range(len(protein) - kd_window + 1):
    window = protein[i:i+kd_window] # slicing the window
    value = 0 # initializing the value
    for aa in window: # looping over each amino acid in the window
        value += kd[aa] # adding the hydrophobicity value from the scale
    value /= kd_window # dividing by the window size to get the mean
    kd_value.append(value) # appending to the list of values

# Assigning helix regions based on the threshold
start = None # initializing the start index
for i in range(len(kd_value)): # looping over each value
    if kd_value[i] > kd_threshold: # if the value is above the threshold
        if start == None: # and if we have not started a helix region
            start = i # set the start index to the current index
    else: # if the value is below or equal to the threshold
        if start != None: # and if we have started a helix region
            end = i + kd_window - 1 # set the end index to the current index plus
window size minus one
            kd_helix.append((start, end)) # append the start and end indices as a
tuple to the list of helix regions
            start = None # reset the start index to None

# Printing the results
```

```
print("--------------------------------")

print("Protein sequence:", protein.seq)
print("--------------------------------")

print("Amino acid composition in the protein sequence:")
for aa, percent in aa_comp.items():
    print(aa, percent*100)
print("--------------------------------")

print("Molecular weight of the protein:", mw, "Da")
print("--------------------------------")

print("Predicted helix regions using Kyte-Doolittle scale:")
for start, end in kd_helix:
    print("From", start, "to", end)
```

Output for the protein Aspartate kinase protein fasta file:

C:\Users\DELL\OneDrive\Desktop\bio computing>python -u "c:\Users\DELL\OneDrive\Desktop\bio computing\onezafinalproject.py"

ONEZA HASSAN ALVI

REG NUM: 04281913032

BIOCOMPUING FINAL PROJECT

Enter the accurate protein fasta file name from the working directory or give the whole path: C:\Users\DELL\OneDrive\Desktop\bio computing\Aspartate_kinase.fasta

---

Protein sequence:
MGLIVQKFGGTSVGSVEKILNVANRVIEEKQRGHDVVVVVSAMGKSTDSLVALAKEITEQPSSREMDMLLTTGEQVTIS
LLTMALQNKGYDAVSYTGWQAGIETENIHGNARITNIDTAKLKERLNEGKIAVVAGFQGVTAEGEITTLGRGGSDTTAV
ALAAALKADKCDIYTDVPGVFTTDPRYVKTARKLAGISYDEMLELANLGAGVLHPRAVEFAKNYQVPLEVRSSTEKEAGT
LIEEESSMEQNLIVRGIAFEDQITRVTVCGLASGLTTLSTIFTTLAKQNINVDIIIQSVTGTSKTSISFSVKTEDLKRTVEVLEEY
KDGLDYEQIETENRLAKVSIVGSGMISNPGVAAEMFAVLAEKNIQVKMVSTSEIKVSTVVNEGDMVKAVEALHDAFELS
KQTAVI

Amino acid composition in the protein sequence:

A 9.04645476772616

C 0.4889975550122249

D 4.400977995110025

E 9.04645476772616

F 2.2004889975550124

G 7.823960880195599

H 0.9779951100244498

I 7.090464547677261

K 6.356968215158925

L 8.06845965770171

M 2.689486552567237

N 3.9119804400977993

P 1.466992665036675

Q 3.6674816625916873

R 3.4229828850855744

S 7.090464547677261

T 9.290953545232274

V 10.757946210268948

W 0.24449877750611246

Y 1.9559902200488997

Molecular weight: 43930.56460000005 Da

Predicted helix regions using Kyte-Doolittle scale:

From 35 to 46

From 74 to 86

From 129 to 140

From 155 to 166

From 353 to 364