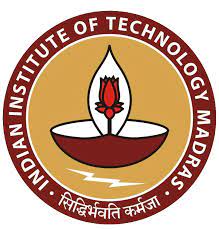
..

**BT 3040: Bioinformatics**

**Assignment 9**



Indian Institute of Technology Madras

Atharva Mandar Phatak | BE21B009

Department of Biotechnology

## **Q1**) **1. Identify the pair of sequences which are close to each other using Hamming and Euclidean distance methods.**

## **(i)AMENLNMDLLYMAAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVG**

## **LGLYVMFAVA**

## **(ii)AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAP**

## **VFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPN**

## **DGIESYSLFYKIPI**

## **(iii)MALLPAAPGAPARATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEY**

## **LAALKQKSRHAA**

#Q1 | Atharva Mandar Phatak | BE21B009 | Assignment 9 | BT3040 |

def calculate\_aa\_percentage(sequence):

    # Define the 20 standard amino acids

    amino\_acids = "ACDEFGHIKLMNPQRSTVWY"

    # Initialize a dictionary to store the percentage occurrence of each amino acid

    aa\_percentage = {aa: 0 for aa in amino\_acids}

    # Count occurrences of each amino acid in the sequence

    total\_aa\_count = 0

    for aa in sequence:

        if aa in amino\_acids:

            aa\_percentage[aa] += 1

            total\_aa\_count += 1

    # Calculate percentage occurrence of each amino acid

    if total\_aa\_count > 0:

        for aa in aa\_percentage:

            aa\_percentage[aa] = (aa\_percentage[aa] / total\_aa\_count) \* 100

    # Sort the dictionary by keys

    sorted\_aa\_percentage = {k: v for k, v in sorted(aa\_percentage.items())}

    return sorted\_aa\_percentage

def hamming(dict1, dict2):

    # Initialize the sum of differences

    sum\_of\_differences = 0

    # Iterate through each key in the dictionaries

    for key in dict1.keys():

        # Calculate the difference between the values of corresponding keys

        difference = dict1[key] - dict2[key]

        # Take the modulus of the difference

        difference\_mod = abs(difference)  # Assuming you meant mod 100

        # Add the modulus to the sum

        sum\_of\_differences += difference\_mod

    return sum\_of\_differences

def euclidean(dict1, dict2):

    # Initialize the sum of squared differences

    sum\_of\_squared\_differences = 0

    # Iterate through each key in the dictionaries

    for key in dict1.keys():

        # Calculate the difference between the values of corresponding keys

        difference = dict1[key] - dict2[key]

        # Square the difference

        squared\_difference = difference \*\* 2

        # Add the squared difference to the sum

        sum\_of\_squared\_differences += squared\_difference

    # Take the square root of the sum of squared differences

    sqrt\_sum = (sum\_of\_squared\_differences)\*\*(0.5)

    return sqrt\_sum

dict1 = calculate\_aa\_percentage(seq1)

dict2 = calculate\_aa\_percentage(seq2)

dict3 = calculate\_aa\_percentage(seq3)

hammering\_1\_2 = hamming(dict1, dict2)

hammering\_1\_3 = hamming(dict1, dict3)

hammering\_2\_3 = hamming(dict2, dict3)

eucledian\_1\_2= euclidean(dict1, dict2)

eucledian\_1\_3= euclidean(dict1, dict3)

eucledian\_2\_3= euclidean(dict2, dict3)

print(f"For pair 1 and 2, Hamming Distance is {hammering\_1\_2}")

print(f"For pair 1 and 3, Hamming Distance is {hammering\_1\_3}")

print(f"For pair 2 and 3, Hamming Distance is {hammering\_2\_3}")

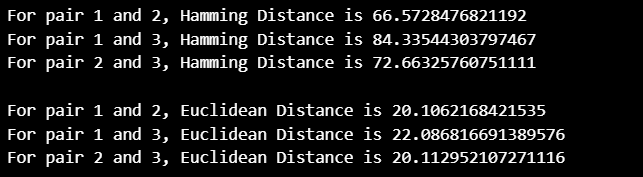
print("")

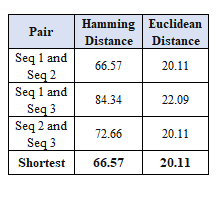
print(f"For pair 1 and 2, Euclidean Distance is {eucledian\_1\_2}")

print(f"For pair 1 and 3, Euclidean Distance is {eucledian\_1\_3}")

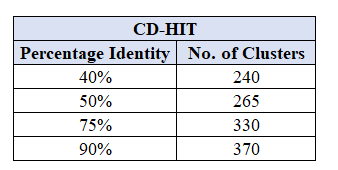
print(f"For pair 2 and 3, Euclidean Distance is {eucledian\_2\_3}")

Output:





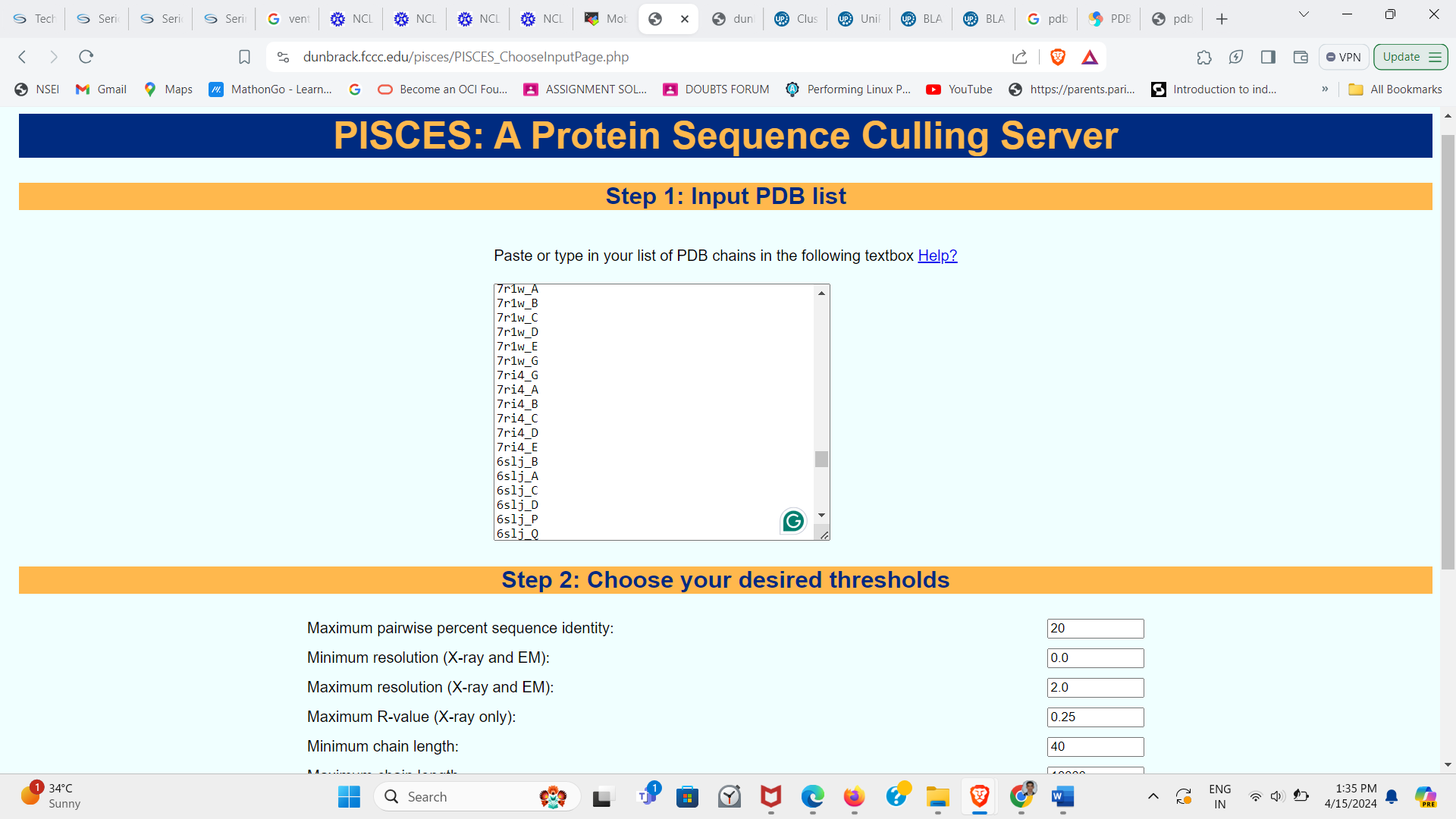
## **Q2) Get the non-redundant sequences of beta barrel membrane proteins with sequence identities of less than 40%, 50%, 75% and 90% using CD-HIT**

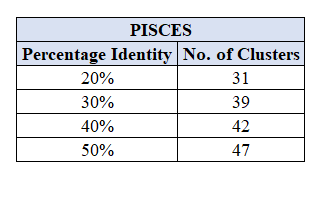


*The downloaded files are in the folder*

## **Q3) Get the non-redundant sequences of the same type of proteins with sequence identities of less than 20%, 30%, 40% and 50% using PISCES (https://dunbrack.fccc.edu/pisces/**

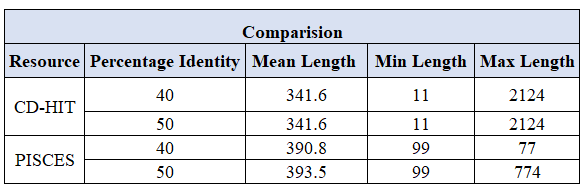
1. 20% (similarly for others, screenshot attaching only for one)





*The downloaded files are in the folder*

### **Q4) Compare the results obtained with the cut-offs 40% and 50%.**



The number of chains for **40%** and **50%** are **42** and **47** respectively

Analysis:

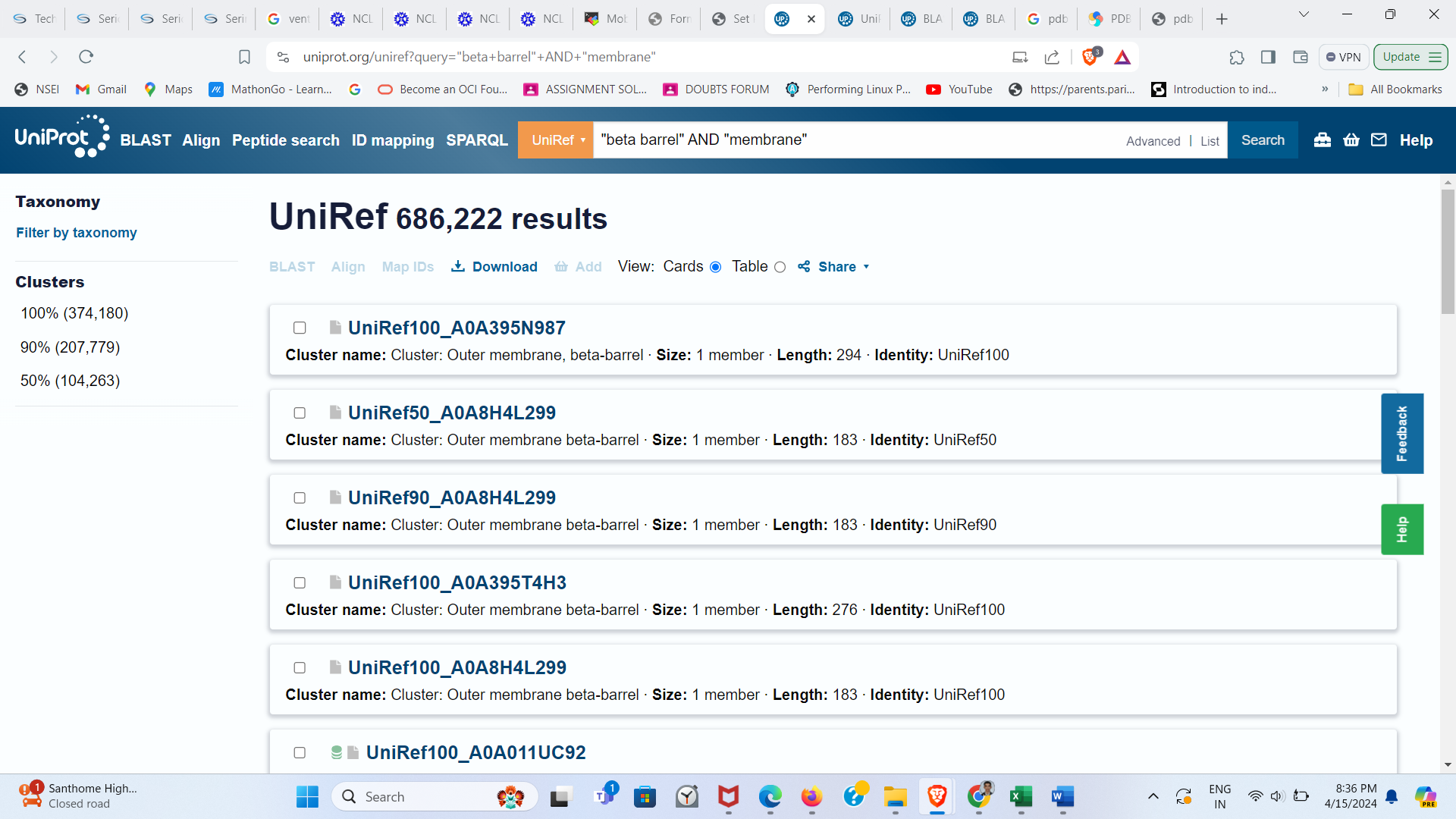
Comparing the length of sequences between PISCES and CD-HIT datasets reveals that, on average, PISCES tends to include longer sequences, possibly favouring complete protein structures, while CD-HIT encompasses a broader range of sequence lengths, from short fragments to longer ones.

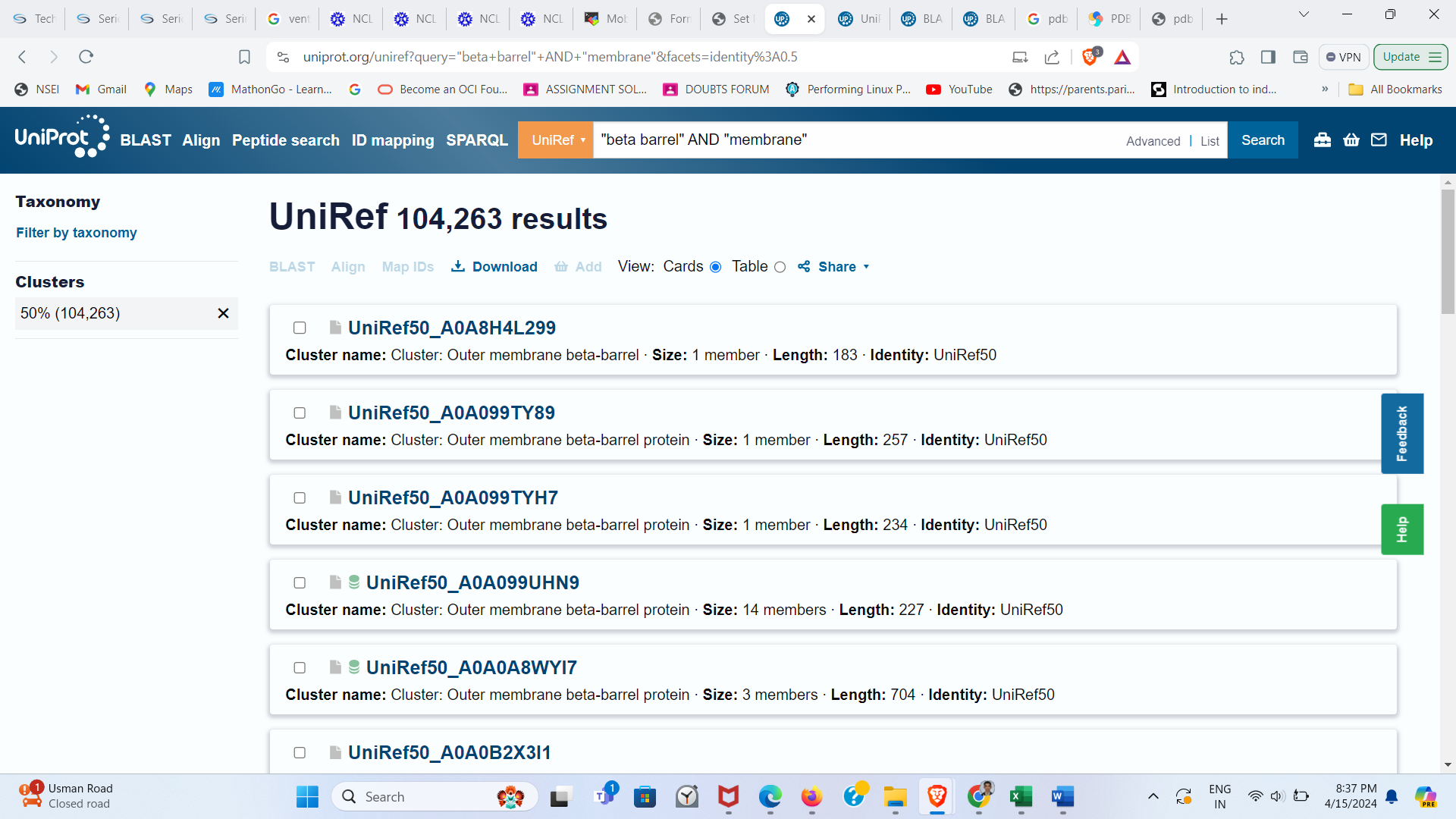
CD-HIT clusters formed with default cutoff values of 40% and 50% for redundant beta barrel membrane proteins resulted in smaller clusters with higher sequence similarity, suggesting tight groupings of very similar sequences.

PISCES selects sequences above a higher cutoff based on structural quality and non-redundancy, indicating a focus on structural integrity. The resolutions of sequences in PISCES datasets show minimal variation between the 40% and 50% cutoffs, indicating consistently high-quality structures.

In terms of unique contributions, CD-HIT is well-suited for studies requiring high sequence similarity, such as sequence homology or evolutionary analysis, while PISCES is more beneficial for structural biology, prioritizing high-quality and unique protein structures for applications like modelling and drug design.

## **Q5) Extract the data with the cut-off of 50% from UniProt and compare with CD-HIT and PISCES**





Comparison:

