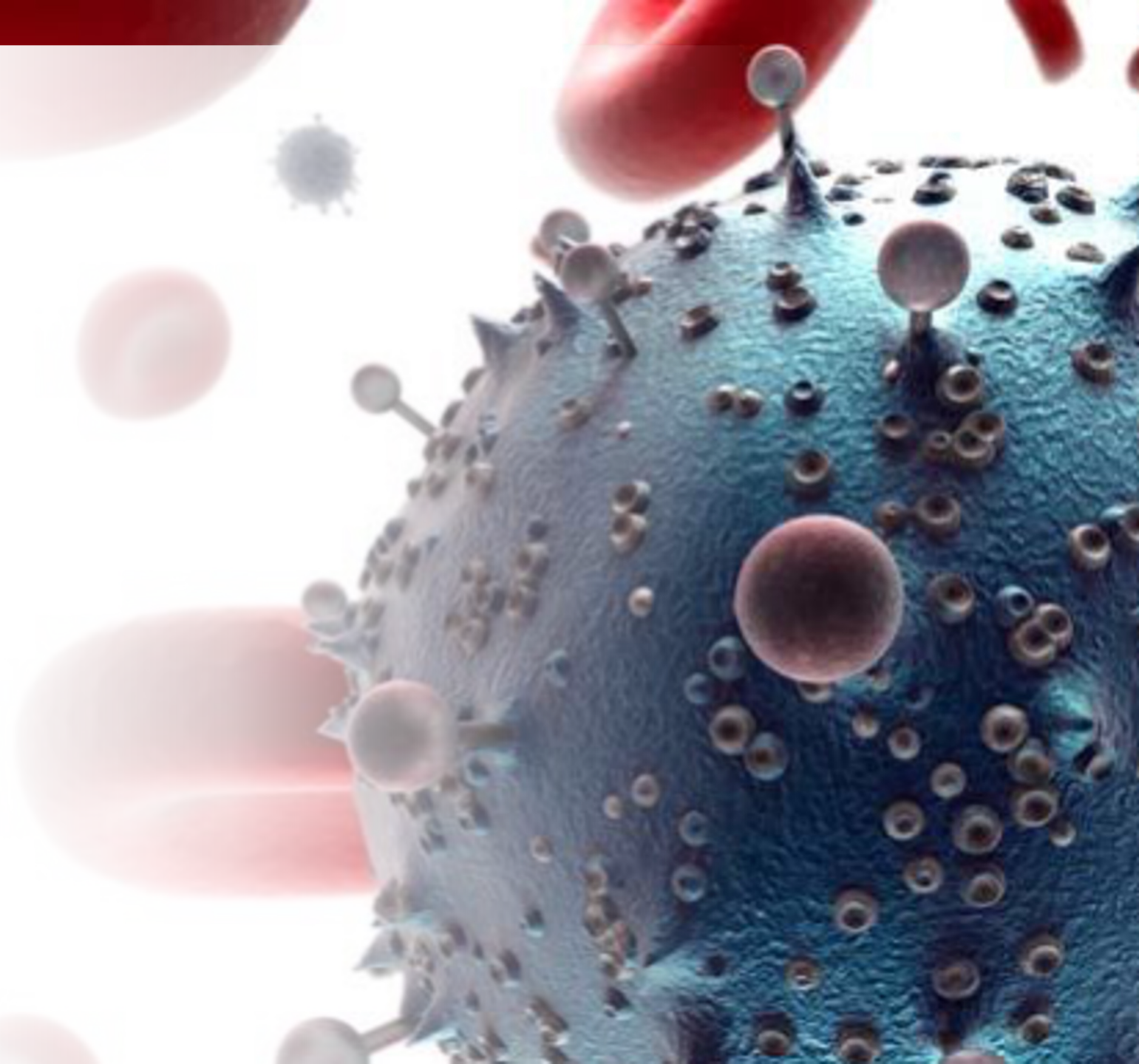
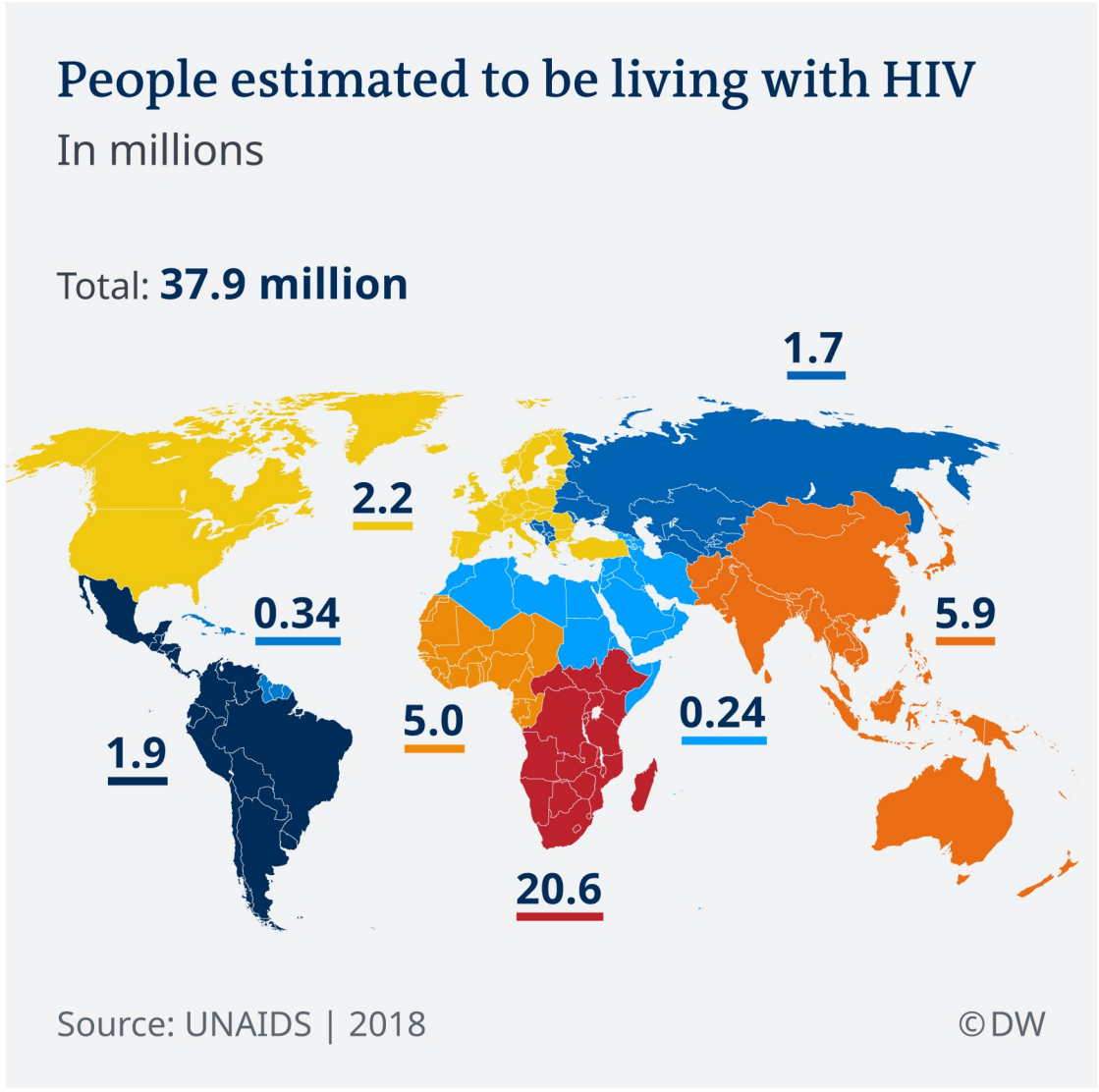
Comparative computational Analysis of HIV-1 sequence variants in Africa and Europe

Presented By
*Mohamed Elmanzalawi
*Sherouk Mahmoud
*Ahmed Elghamry



HIV caused 34 million deaths



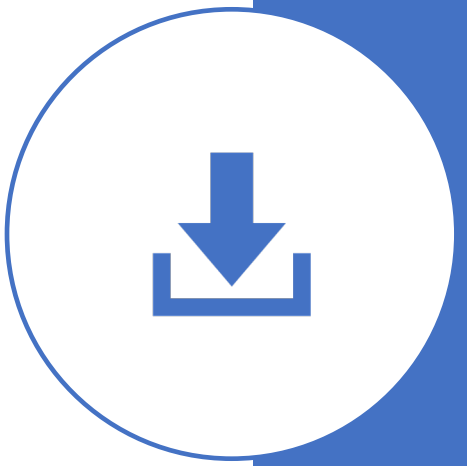
Agenda

Data Collection	Alignment	Consensus sequence	Dissimilar region extraction
Phylogenetic Tree	CG content	Functional Products	Flash Back Functions Dissimilar Regions
second Function CG content	Dissimilar Regions	Conclusion • This study was performed on the same virus, HIV-1 but it shows that the environment has an effect in the virus development as by changing the continent the whole genome of the virus got different nucleotides which may represent different functional products at the end as shown in result. The more critical this functional products to the virus, the more the notable effect on its behavior and transmit.	

Data Collection

Data Collection

- print('please enter the location of the following files.')
- Download_seq = input('1- Desired folder to keep all the results’)
- Alignment_exe = input('2- Clustalw2.exe folder example:(F:\Files\Clustral W) ')
- First_Accession_List = input('please enter your first accession list ‘
- 'example:(MZ767421.1,MZ767468.1) ').split(',')
- Second_Accession_List = input('please enter your second accession list').split(',')



Files creation

- clustalw_exe = "%s\clustalw2.exe" % Alignment_exe
- First_Alignment = '%s\Clustal_Alignment.ALN' % Download_seq
- second_Alignment = '%s\second_Alignment.ALN' % Download_seq
- consensus_Sequences = '%s\consensus_Sequences.fasta' % Download_seq
- Third_Alignment = '%s\Main_Alignment.ALN' % Download_seq
- Final_Alignment = '%s\Final_Alignment.ALN' % Download_seq
- CG_content = '%s\CG_content.txt' % Download_seq
- Dissimilar_Positions_txt = '%s\Dissimilar_Positions.txt' % Download_seq

Downloading sequences by Entrez.efetch

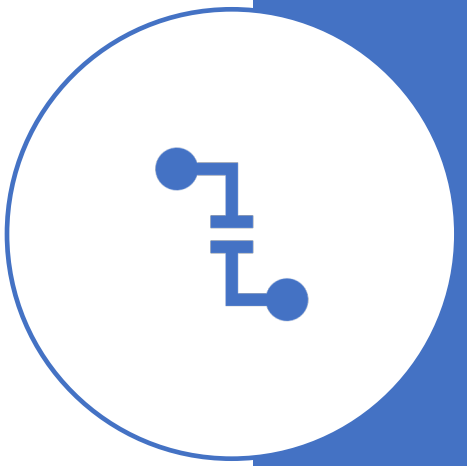
- First_Sequences = "%s\First_Sequences.fasta" % Download_seq
- Entrez.email = 'medo611@hotmail.com'
- net_handle = Entrez.efetch(
- db="nucleotide", id=First_Accession_List, rettype="fasta", retmode="text")
- out_handle = open(First_Sequences, "w")
- out_handle.write(net_handle.read())
- out_handle.close()
- net_handle.close()

Alignment

Alignment

- clustalw_cline = ClustalwCommandline(clustalw_exe, infile=First_Sequences, outfile=First_Alignment)
- clustalw_cline()

- # saving the results in separate files
- Alignment_1_list = []
- Alignment_2_list = []
- for seq_record in SeqIO.parse(First_Alignment, format='clustal'):
- Alignment_1_list.append(seq_record.seq)
- for seq_record in SeqIO.parse(second_Alignment, format='clustal'):
- Alignment_2_list.append(seq_record.seq)



Consensus sequence

Consensus sequence

- consensus_Seq_file_in = open(consensus_Sequences, "w")
- consensus_Seq_1 = **Function_Consesus.consensus**(Alignment_1_list)
- consensus_Seq_2 = Function_Consesus.consensus(Alignment_2_list)
- consensus_Seq_file_in.write(">The_First_list_Consensus_Sequence" + "\n" + consensus_Seq_1 +
- '\n\n' + ">The_Second_list_Consensus_Sequence" + "\n" + consensus_Seq_2)
- consensus_Seq_file_in.close()

Dissimilar region extraction

Dissimilar region

Dissimilar_Positions - Notepad

File Edit Format View Help

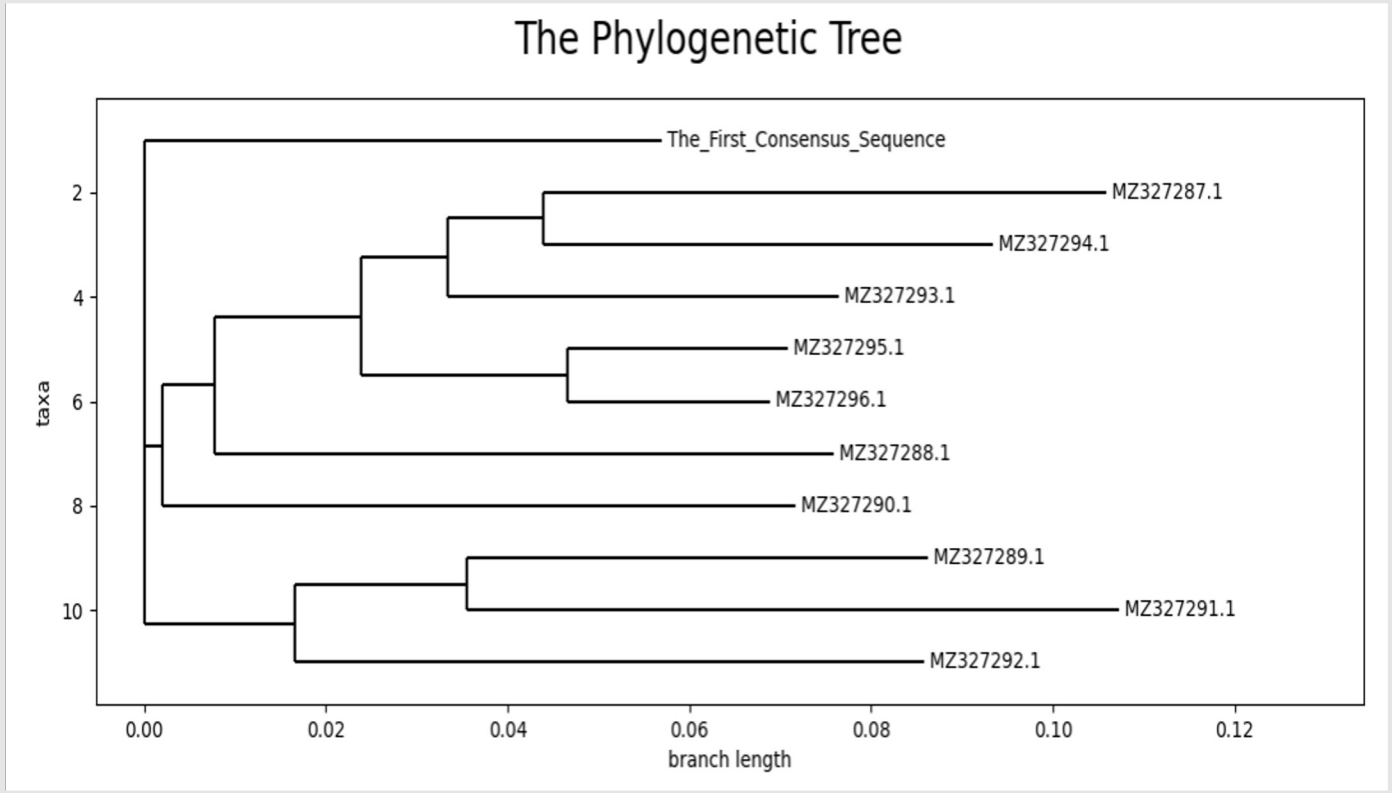
```
There is a dissimilar region at index 770
Which is G in The_Second_list_Consensus_Sequ and in the Main consensus  is A
There is a dissimilar region at index 772
Which is T in The_Second_list_Consensus_Sequ and in the Main consensus  is C
There is a dissimilar region start at index 774 and end at index 775
Which is CA in The_Second_list_Consensus_Sequ and in the Main consensus  is --
There is a dissimilar region at index 781
Which is A in The_Second_list_Consensus_Sequ and in the Main consensus  is G
There is a dissimilar region at index 794
Which is A in The_Second_list_Consensus_Sequ and in the Main consensus  is G
There is a dissimilar region at index 798
```

Phylogenetic Tree

Phylogenetic tree preparation

```
Final_Fasta = '%s\Final_Fasta.fasta' % Download_seq
Final_Fasta_in = open(Final_Fasta, 'w')
Second_Sequences_in = open(Second_Sequences, 'r')
Final_Fasta_in.write(">The_First_CS" + "\n" + consensus_Seq_1
+ '\n\n' + Second_Sequences_in.read())
Second_Sequences_in.close()
Final_Fasta_in.close()
```

Drawing phylogenetic tree



- dnd_file = ("%s\Final_Fasta.dnd" % Download_seq)
- tree = Phylo.read(dnd_file, "newick")
- fig_2 = plt.figure(figsize=(13, 5), dpi=100) # create figure & set the size
- fig_2.suptitle('The Phylogenetic Tree',

  fontsize=20)
- axes = fig_2.add_subplot(1, 1, 1)
- Phylo.draw(tree, axes=axes)
- fig_2.savefig("%s\The Phylogenetic Tree" % Download_seq) # saving it for the user

CG content

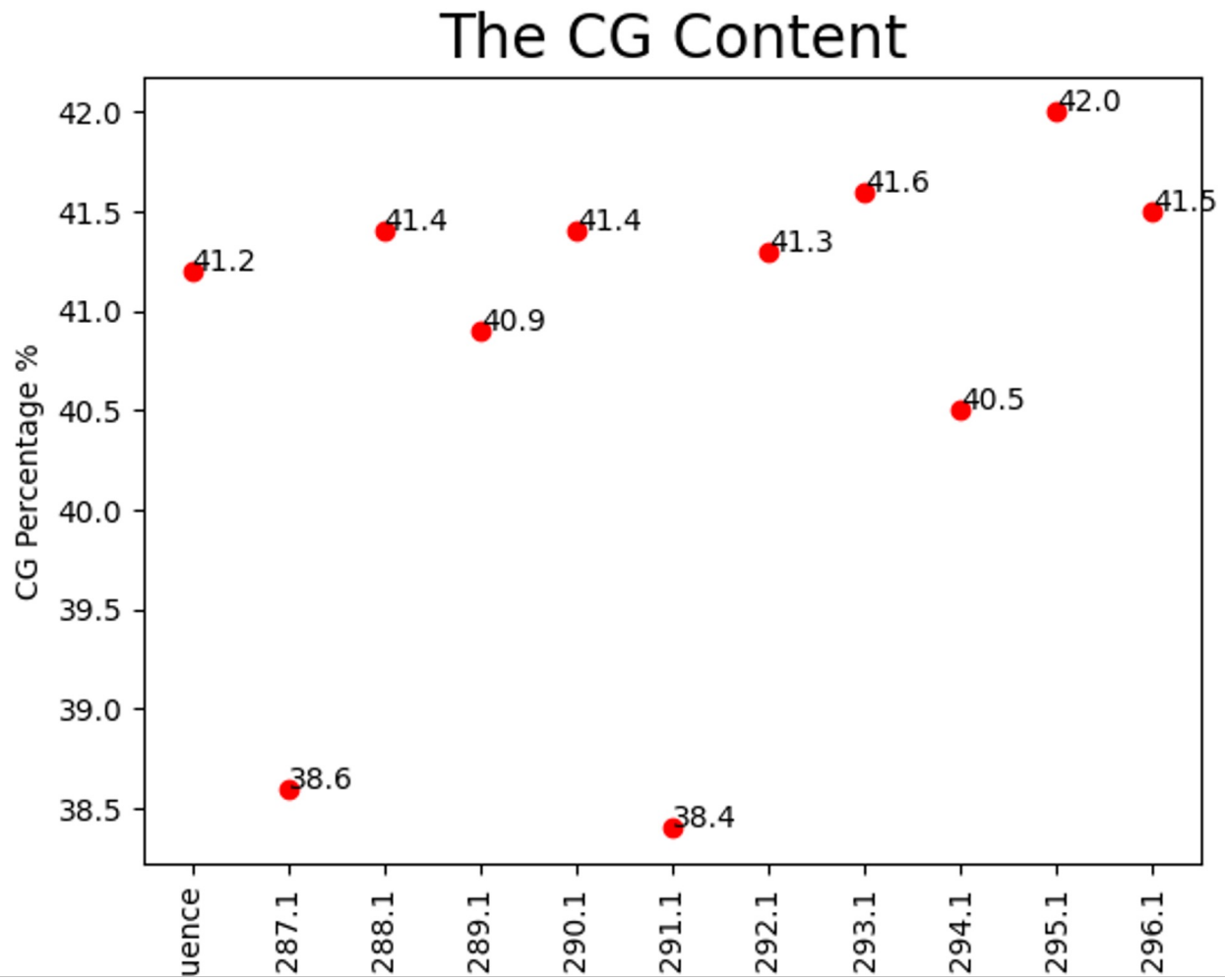
CG content

```
IDs = []
CG_percentage = []
CG_content_open = open(CG_content, 'w')
for Sequence in SeqIO.parse(Final_Fasta, format='fasta'):
    IDs.append(Sequence.id)
    CG_percentage.append(Function_CG_Content.cg_content(Sequence.seq))
    CG_content_open.write('The sequence %s has CG content= %f \n'
                          % (Sequence.id, Function_CG_Content.cg_content(Sequence.seq)))
CG_content_open.close()
```

Scatter plot of CG content

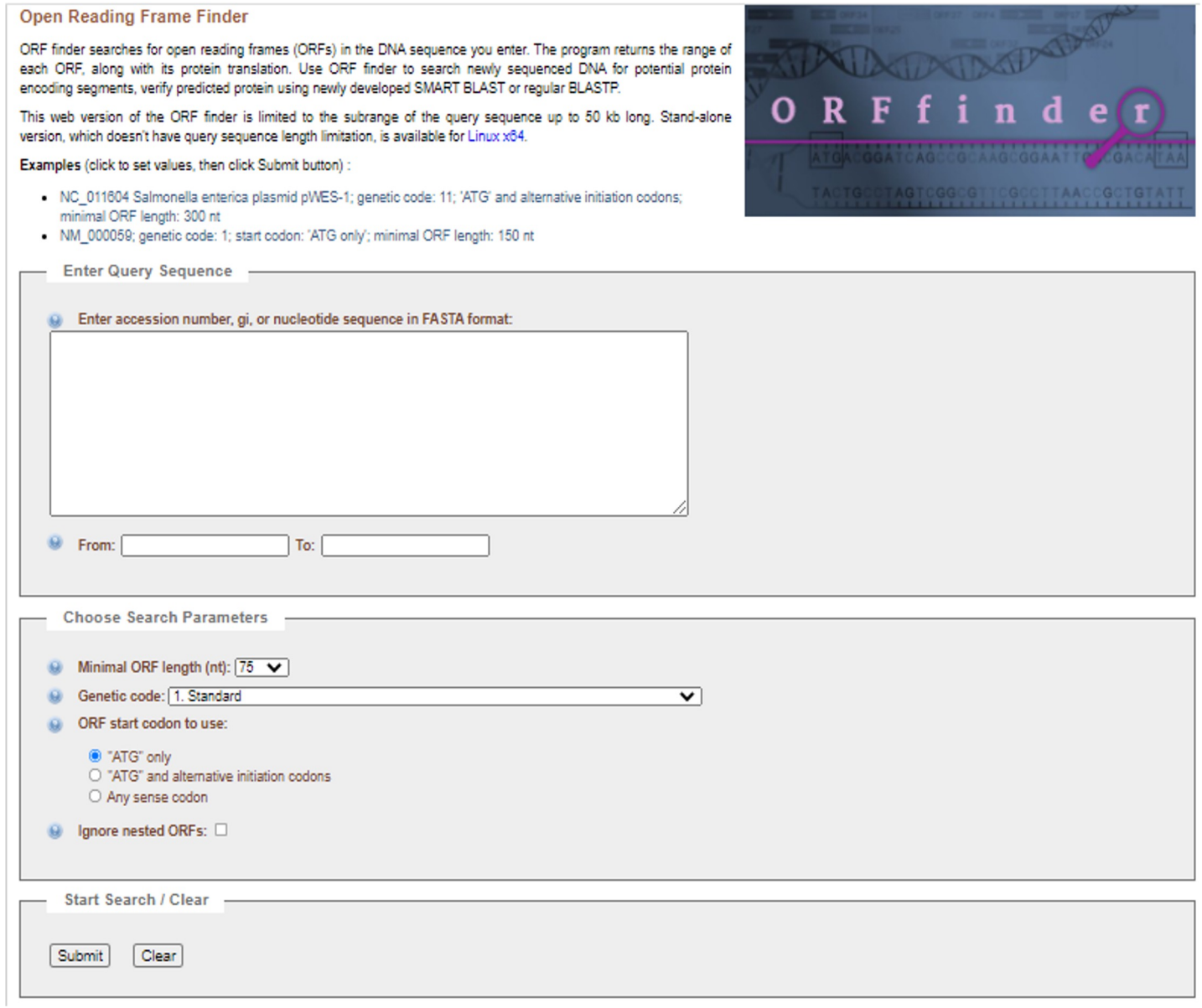
- plt.title('The CG Content', fontsize=20)
- plt.scatter(x=IDs, y=CG_percentage, marker='o', color='r')

- for i, txt in enumerate(CG_percentage):
- plt.annotate(txt, (IDs[i], CG_percentage[i]))
- plt.ylabel('CG Percentage %')
- plt.xticks(rotation='vertical')
- plt.savefig("%s\The CG Content scatter plot" % Download_seq)



Functional Products

Predicting the ORF locations



Open Reading Frame Finder

ORF finder searches for open reading frames (ORFs) in the DNA sequence you enter. The program returns the range of each ORF, along with its protein translation. Use ORF finder to search newly sequenced DNA for potential protein encoding segments, verify predicted protein using newly developed SMART BLAST or regular BLASTP.

This web version of the ORF finder is limited to the subrange of the query sequence up to 50 kb long. Stand-alone version, which doesn't have query sequence length limitation, is available for Linux x64.

Examples (click to set values, then click Submit button) :

- NC_011604 Salmonella enterica plasmid pWES-1; genetic code: 11; 'ATG' and alternative initiation codons; minimal ORF length: 300 nt
- NM_000059; genetic code: 1; start codon: 'ATG only'; minimal ORF length: 150 nt

Enter Query Sequence

Enter accession number, gi, or nucleotide sequence in FASTA format:

From: To:

Choose Search Parameters

Minimal ORF length (nt): 75

Genetic code: 1. Standard

ORF start codon to use:

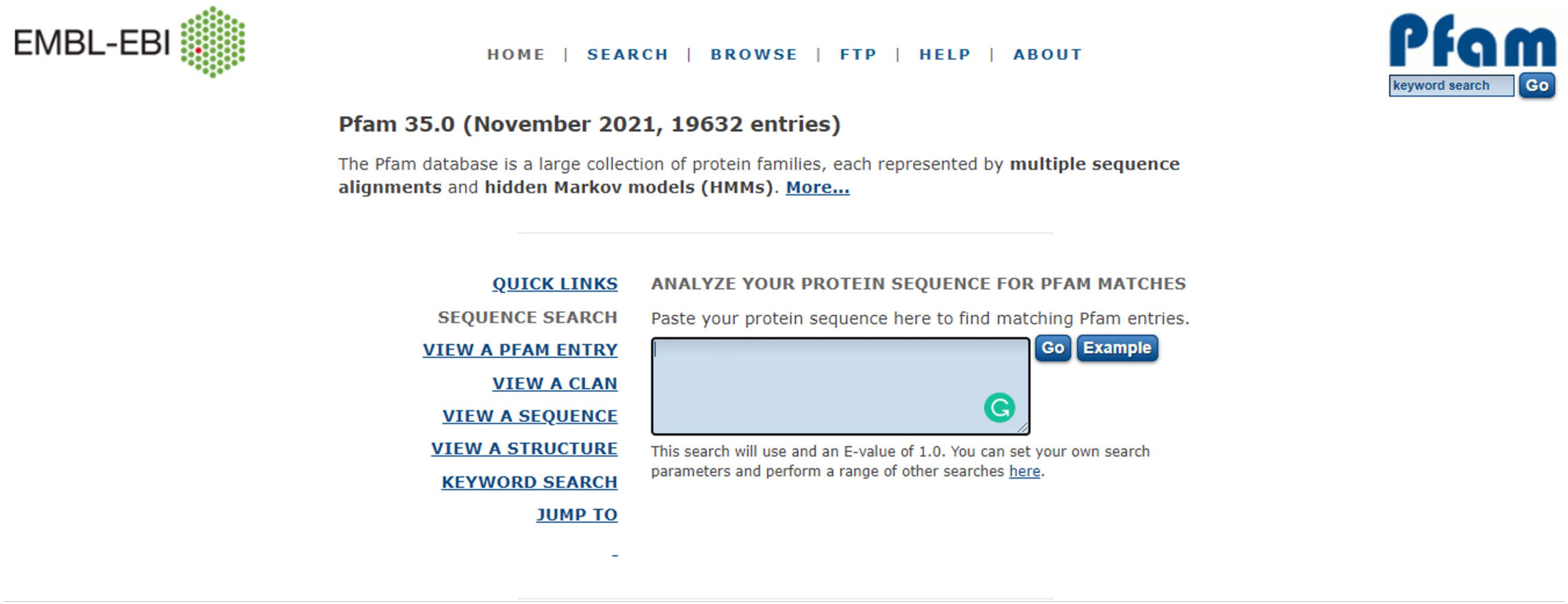
- "ATG" only
- "ATG" and alternative initiation codons
- Any sense codon

Ignore nested ORFs:

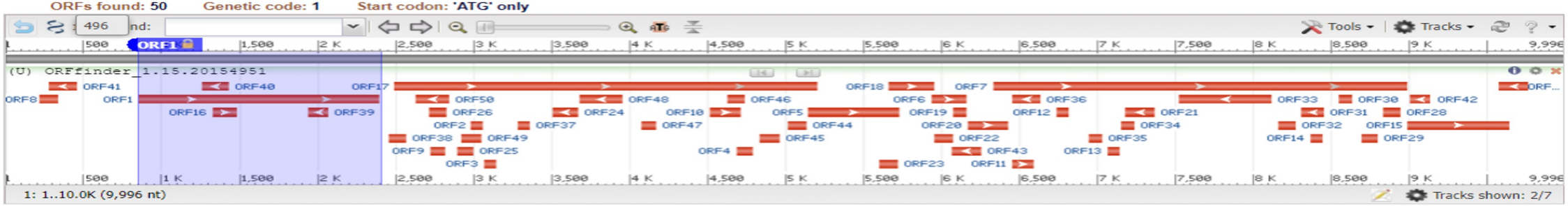
Start Search / Clear

Submit Clear

Predicting the functional product



ORFs found: **50** Genetic code: **1** Start codon: **'ATG' only**



Six-frame translation...

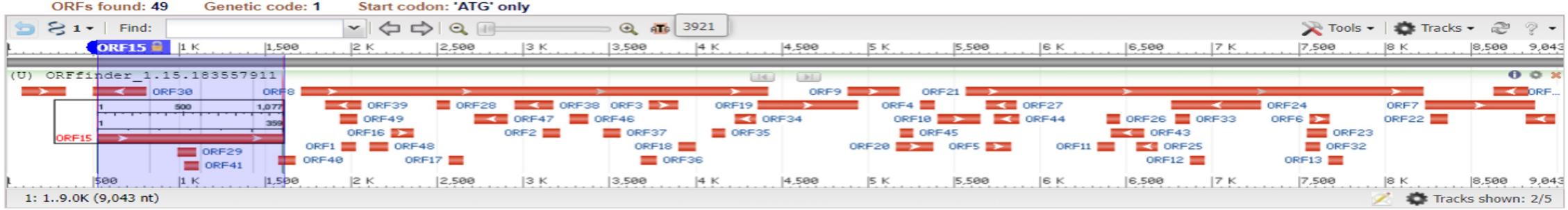
ORF1 (515 aa) Display ORF as... Mark

```
>lcl|ORF1
MGARASVLSGGKLDEWEKIRLRPGGKKKYKLKHIVWASRELERFALNPGL
LETSEGCKQIIEQLQPAIKTGTEELKSLYNTVATLYCVHERIKVADTKEA
LDKIEEEQNKSKKKAQQATADTRNADTKNSSQVSQNYPIVQNLQGQMVHQ
ALSPRTLNAWVKVIEEKAFSPEVIPMFTALSEGATPQDLNTMLNTVGGHQ
AAMQMLKETINEEAAEWDRLHPVQAGPIAPGQMREPRGSDIAGTTSTLQE
QIAWMTSNPPIPVGEIYKRWIILGLNKIVRMYSPVSILDIRQGPKEPFRD
YVDRFYKTLRAEQASQDVKNWMTETLLVQNANPDCKTILKALGPAATLEE
MMTACQGVGGPGHKARVLAEAMSQATNRNAAIMMQRGNFKGPKRIIKCFN
CGKEGHIAKNCRAPRKKGCWKCGKEGHQMKDCTERQANFLGKIWPSHKGR
PGNFLQSRPEPTAPPAESFGFGEEITPSQKQEQKDPSQEQEQKNKELYPP
LASLKSLFGNDPSSQ
```

Mark subset... Marked: 0 Download marked set as Protein FASTA

Label	Strand	Frame	Start	Stop	Length (nt | aa)
ORF17	+	3	2496	5204	2709 | 902
ORF7	+	1	6337	8991	2655 | 884
ORF1	**+**	**1**	**853**	**2400**	**1548 | 515**
ORF15	+	2	8993	9643	651 | 216
ORF33	-	2	8117	7527	591 | 196

ORFs found: **49** Genetic code: **1** Start codon: **'ATG' only**



Six-frame translation...

ORF15 (358 aa) Display ORF as... Mark

```
>lcl|ORF15
MTHQSMSPRTLNAWVKVIEEKAFSPEVIPMFSALSEGATPQDLNMMLNIV
GGHQAAMQMLKDTINEEAAEWDRIHPVQAGPIPPGQMREPRGSDIAGTTS
TLQEQIGWMTSNPPIPVGEIYKRWIILGLNKIVRMYSPVSILDIKQGPKE
PFRDYVDRFFKTLRAEQATQEVKNWMTETLLVQNANPDCKTILRALGPGA
TLEEMMTACQGVGGPGHKARVLAEAMSQVQQSGTAIMMMQRGNFRGQRRI
IKCFNCGKEGHLARNCKAPRKKGCWKCGKEGHQMKDCTERQANFLGKIWP
SNKGRPGNFPQSRPEPTAPPAESLGMGEEIASSPKQEPRDKELYPPLTSL
KSLFGNDP
```

Mark subset... Marked: 0 Download marked set as Protein FASTA

Label	Strand	Frame	Start	Stop	Length (nt | aa)
ORF15	**+**	**3**	**528**	**1604**	**1077 | 358**
ORF7	+	1	8230	8871	642 | 213
ORF19	+	3	4362	4943	582 | 193
ORF24	-	1	7279	6764	516 | 171

Absence of gag protein

The first consensus sequence (Africa)

Family	Description
Gag_p17	gag gene protein p17 (matrix protein)
Gag_p24_C	Gag protein p24 C-terminal domain
Gag_p6	Gag protein p6
zf-CCHC	Zinc knuckle
zf-CCHC	Zinc knuckle
Gag_p24	gag protein p24 N-terminal domain

The second consensus sequence

Family	Description
Gag_p24_C	Gag protein p24 C-terminal domain
Gag_p6	Gag protein p6
zf-CCHC	Zinc knuckle
zf-CCHC	Zinc knuckle

The 1st Consensus sequence of Africa	ORF17	ORF5	ORF7
The 2nd Consensus sequence of Europe	ORF8	ORF19	ORF21
No. of dissimilar regions	177	41	207

Flash Back

Functions

Dissimilar Regions

Functions

First Function : Consensus Sequence

Creating function to get consensus sequence

```
def consensus (Alignment_list):
    import numpy as np
    import textwrap
```

first we will create a list for each nucleotide following by another two lists (one for collecting all list of nucleotide and other for consensus sequence).

```
    C=[ ]
    G=[ ]
    T=[ ]
    A=[ ]
    ALL=[ ]
    consensus = [ ]
```

Making array from alignment list

We will take alignment list from the user and convert it to array (this is why we import numpy package) and rows of array will be the number of sequences and columns will be the number of nucleotides in any sequence in the list since they are all the same.

M=np.array(Alignment list).reshape(len(Alignment_list), len(Alignment_list[0]))

Now we will for loop to loop over every column along the range of length of any sequence

```
for base in range (len(Alignment_list[0])
    A_count=0
    C_count= 0
    G_count=0
    T_count= 0
# we made counter for every nucleotide.
```

we will loop in every column changing only the y axis=base every iteration

```
  for NUC in M[:,base]:
#Now if we find any of nucleotides in the entire column we will add it to the counter
        if NUC ==“A” :
                A_count += 1
        elif NUC ==“C” :
                C_count += 1
        elif NUC == “G”
                G_count +=1
        elif NUC ==“T”
                T_count  +=1
```

Now we will append the value of each counter to its respective list

```
A.append(A_count )
C.append(C_count)
G.append(G_count )
T.append (T_count)
```

```
# Now we will collect each lists in one list (All list )

All.append(A)

All.append(C)

All.append(G)

All.append(T)

#Now we will change our All list to a matrix which has 4 rows will represent the count of A,C,G,T respectively at that index.

All_array = np.array(All).reshape(4 , len(A))
```

For NUC in range(len(A)):
we will go column by column using for loop.
if the max value was in the first raw so nucleotide ‘A’ is the dominant and if it was in second raw it will be ‘C’ and if it was in third raw it will be ‘G’ and if it was in fourth raw it will be ‘T’.
This loop keeps going until we cover all the columns.
and after we identify the dominant nucleotide we will append it to our new list consensus.

```
if max(All_array[:,NUC] == All_array[0, NUC]:
        consensus.append(“A”)
elif max(All_array[:,NUC] == All_array[1, NUC]:
        consensus.append(“C”)
elif max(All_array[:,NUC] == All_array[2, NUC]:
        consensus.append(“G”)
elif max(All_array[:,NUC] == All_array[3, NUC]:
        consensus.append(“T”)
```

we will combine our list of nucleotides to form our consensus sequence

combined_consensus = “”.join(consensus)

This step is optional : to make our consensus sequence more appealing to fasta format.

#in our case the line in the fasta file has 70 nucleotide.

This is why we import textwrap package

wrapped=textwrap.fill(combined_consensus, 70)

Return wrapped

```
def consensus(Alignment_list):
    import numpy as np
    import textwrap
    C = []
    G = []
    T = []
    A = []
    All = []
    consensus = []
    M = np.array(Alignment_list).reshape(len(Alignment_list), len(Alignment_list[0]))
    for base in range(len(Alignment_list[0])):
        A_count = 0
        C_count = 0
        G_count = 0
        T_count = 0
        for NUC in M[:, base]:
            if NUC == "A":
                A_count += 1
            elif NUC == "C":
                C_count += 1
            elif NUC == "G":
                G_count += 1
            elif NUC == "T":
                T_count += 1
        A.append(A_count)
        C.append(C_count)
        G.append(G_count)
        T.append(T_count)
    All.append(A)
    All.append(C)
    All.append(G)
    All.append(T)
```

```
All_array = np.array(All).reshape(4, len(A))
for NUC in range(len(A)):
    if max(All_array[:, NUC]) == All_array[0, NUC]:
        consensus.append("A")
    elif max(All_array[:, NUC]) == All_array[1, NUC]:
        consensus.append("C")
    elif max(All_array[:, NUC]) == All_array[2, NUC]:
        consensus.append("G")
    elif max(All_array[:, NUC]) == All_array[3, NUC]:
        consensus.append("T")
combined_consensus = "".join(consensus)
wrapped = textwrap.fill(combined_consensus, 70)
return wrapped
```

second Function CG content

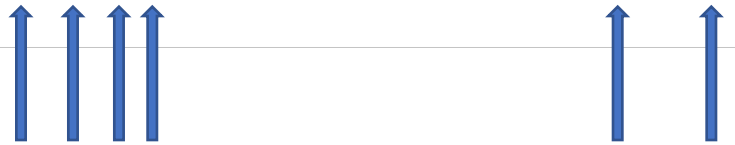
```
def cg_content(Seq) :
    c=Seq.count("C")   # to get number of C in sequence
    g=Seq.count("G")   # to get number of G in sequence
    cg_content=round ( (c + g) / len(seq) * 100 , 1 ) #round to 1st decimal
    return cg_content
```

```python
def cg_content(Seq):
    c = Seq.count('C')  # get the number of C in the sequence
    g = Seq.count('G')  # get the number of G in the sequence
    cg_content = round((c + g) / len(Seq) * 100, 1)  # Rounding to first decimal
    return cg_content
```

Dissimilar Regions

The Dissimilar Region Code

```
The_First_list_Consensus_Seque      CGCAGGACTCGGCTTGCTGAAGCG--CGCACGGCAAGAGGCGAGGGGCAG
The_Second_list_Consensus_Sequ      --------------TGCTGAGGTGCACGCACAGCAAGAGGCGAGAGGCGG
                                               ****** * *  ***** ************ *** *
```



```
Dissimilar_Positions_txt = '%s\Dissimilar_Positions.txt' % Download_seq
Dissimilar_Positions_open = open(Dissimilar_Positions_txt, 'w')
```

- First we will create a path for the txt file that will have our Dissimilar positions And open the file to write the data in it.

```
A = []
y = []
for SeqRecord in AlignIO.read(Third_Alignment, 'clustal'):
   A.append(SeqRecord.seq)
profile = np.array(A)
```

- we will create two empty lists ‘Y’ will capture the index of the dissimilar region and A will contain every sequence in the alignment using for loop .
- Then we will form a matrix using np.array function

```
difference = True
Starting_Gaps = True
```

- Then we will define two Boolean variable
- We will use difference variable so that we print only when difference is false (similar region) so that we get all the dissimilar region before it (difference =True)
- The second variable will help us neglect the first gaps in the beginning of the sequence.

```
for x in range(len(A[0])):
```

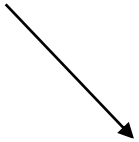
- Using for loop we will go over each column in the matrix and give it 4 conditions.

```
if '-' in profile[:, x] and Starting_Gaps:
    continue
```

- **Condition 1** : This condition was made to neglect the first gaps in the alignment

so if it saw '-' gaps it should neglect the entire column as long as Staring_Gaps==True which will be false only when the script find the first similar region after that this condition will no longer be needed since we have passed the starting gaps.

- Starting gaps == True
- Neglect the gap And continue to the next iteration

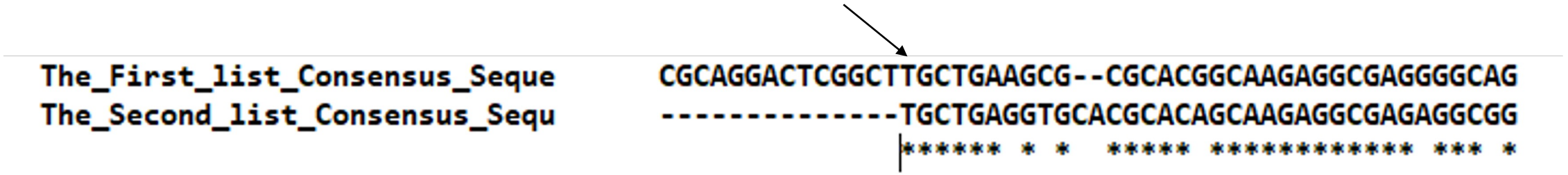


```
The_First_list_Consensus_Seque      CGCAGGACTCGGCTTGCTGAAGCG--CGCACGGCAAGAGGCGAGGGGCAG
The_Second_list_Consensus_Sequ      --------------TGCTGAGGTGCACGCACAGCAAGAGGCGAGAGGCGG
                                                  ****** * *  ***** ********** *** *
```

```
if len(set(profile[:, x])) == 1:
    Starting_Gaps = False
    difference = False
```

- **Condition 2 : Sets neglect repetition, if an entire column contain the same value converted to a set it will have only one value which is our similar region so the set will have one value, So when the length==1 that means this is a similar region and we neglect it and Starting_Gaps = False so that we include gaps in our dissimilar region and not negect it.**

- **Starting gaps == false**
- **difference = False**
- **Since the first condition fulfilled it's purpose the Boolean variable should be false so that we can include gaps in our dissimilar region**

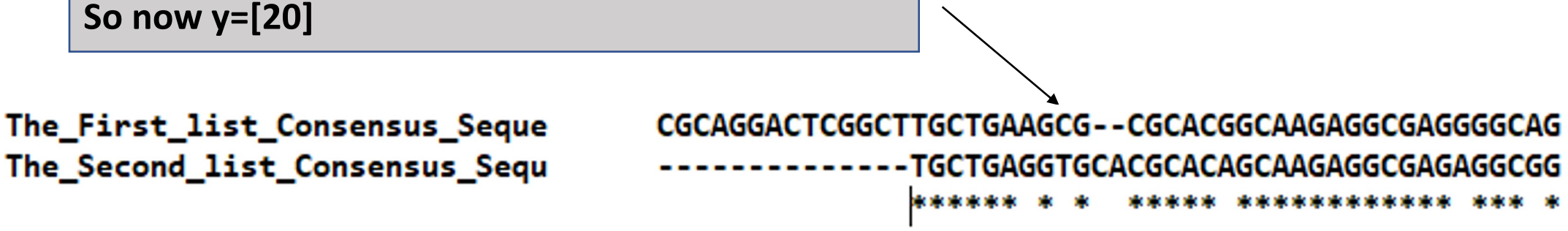


```
if len(set(profile[:, x])) != 1:
    y.append(x)
    difference = True
```

- **Condition 3 :** if len(set) not equal zero so this index has a dissimilar region, so we will add it to the list Y and difference will be True and won't write this value in the file yet since we don't know if the dissimilar region end on that index or not.

- difference = True
- it will add the this index value to the y list

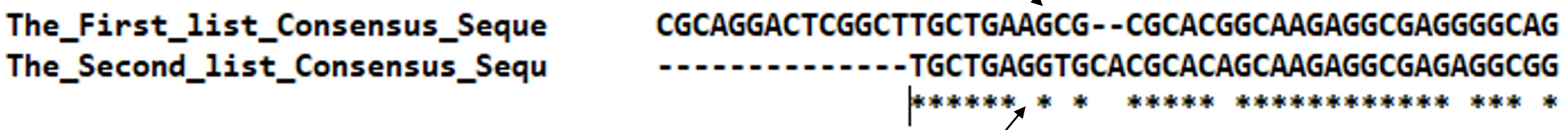
So now y=[20]



if not difference:

- **Condition 4: the script will only get the dissimilar region and print it if the difference is false and the results will depend on the length of y list.**

- difference = False
- So it will starting producing results now



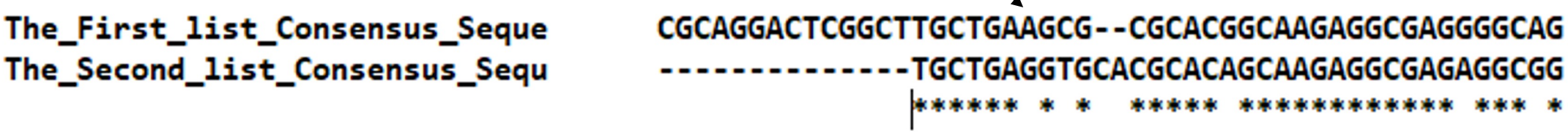
- Y=[20]
- difference = True

if len(y) == 1:

- **if the y list contains only one element that means it is only one index the dissimilarity exists,**

So the script will print the following text.

- **Y=[20]**



```
Dissimilar_Positions_open.write('There is a dissimilar region at index %s \n' % y[0])
for SeqRecord in AlignIO.read(Third_Alignment, 'clustal'):
        if SeqRecord[y[0]] != A[0][y[0]]:
            Dissimilar_Positions_open.write(
                'Which is %s in %s and in the Main consensus  is %s \n' %
                (SeqRecord[y[0]], SeqRecord.id, A[0][y[0]]))
y.clear()
```

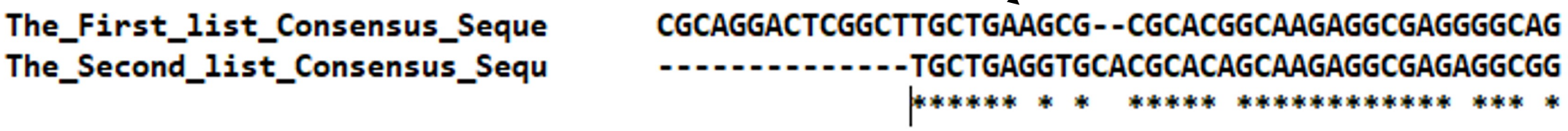
- We will identify the dissimilar index for the user and using for loop we will compare it with the main consensus which is the first sequence in the list A at that index and display the mutated nucleotide.

- **Output:**

```
There is a dissimilar region at index 20
Which is G in The_Second_list_consensus_Sequ and in the Main consensus  is A
```

- Then we will clear y list so that it won't overlap with the next dissimilar region

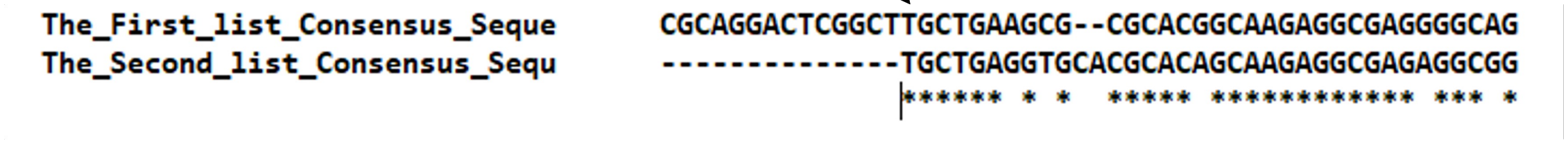
- Y=[20]



```
if len(y) == 0:
    continue
```

- if the list y has no values which may occur if the first part of the sequence has a similarity, so difference will be false and the script will try to print y and this not a dissimilar region and y has no values So this condition was made to continue to the next iteration and skip the rest of the code.

- difference = False
- So it will starting producing results now but y list has no values yet.

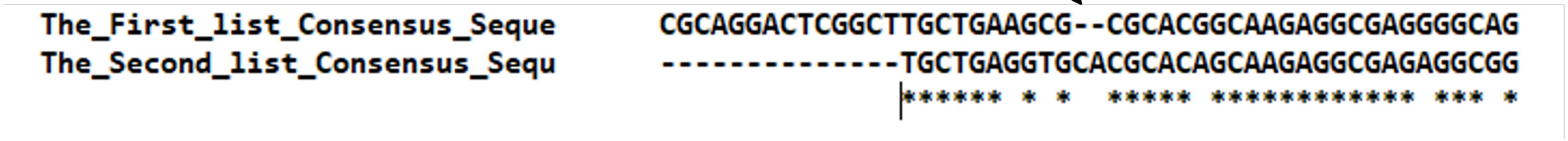


```
if len(y) != 1:
```

- if the y list contains many values that means there is a wide region of dissimilarity exists,

So the script will print the following text.

- Y=[24,25]



```
Dissimilar_Positions_open.write('There is a dissimilar region start at index %s and end at index %s
\n' % (y[0], y[-1]))
        for SeqRecord in AlignIO.read(Third_Alignment, 'clustal'):
                if SeqRecord.seq[y[0]:(y[-1] + 1)] != A[0][y[0]:(y[-1] + 1)]:
                    Dissimilar_Positions_open.write('Which is %s in %s and in the Main consensus  is %s \n'
                                                    % (SeqRecord.seq[y[0]:(y[-1] + 1)], SeqRecord.id, A[0][y[0]:(y[-1] + 1)]))
y.clear()
```

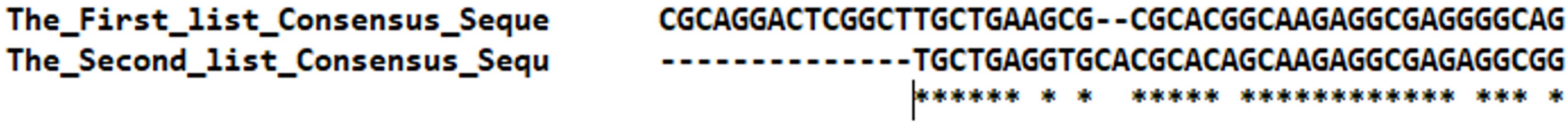
- Same as before the difference is that we will compare range of indexes in every sequence with the main consensus and display the mutated nucleotides and since that last index is not included in range we added 1 to include it .

- **Output:**

'There is a dissimilar region start at index 24 and end at index 25
Which is CA in The_Second_list_consensus_Sequ and in the Main consensus is --

- Then we will clear y list so that it won't overlap with the next dissimilar region

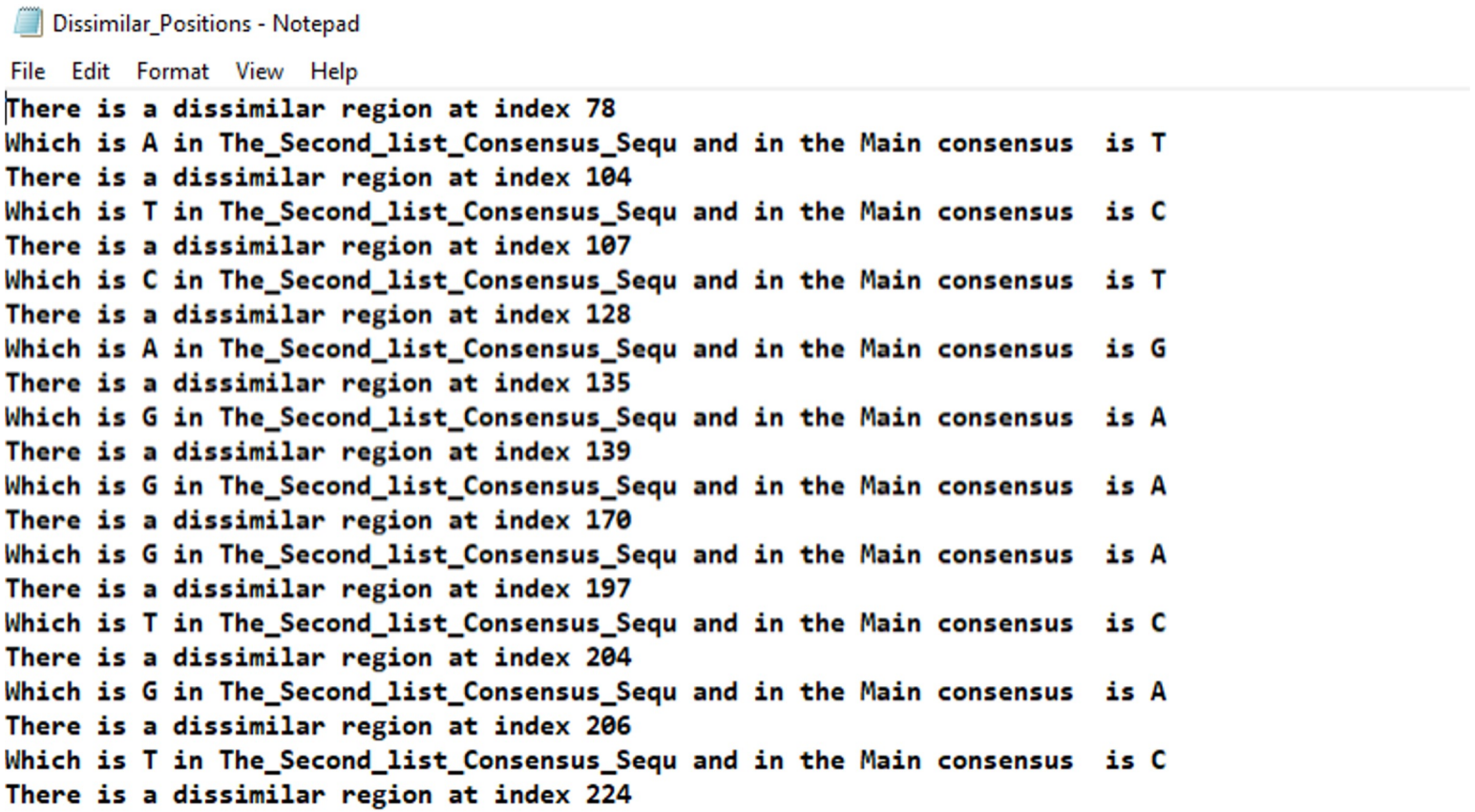
- Y=[24,25]



Dissimilar_Positions_open.close()

- **Then we close the file since we have all the data we need.**

- **The Results will be like this:**



The Full Code part 1

```
# first we will create two empty lists and open a file to store the dissimilar region data.
Dissimilar_Positions_open = open(Dissimilar_Positions_txt, 'w')
A = []
y = []
# List A will capture the sequences in the clustal alignment using for loop.
for SeqRecord in AlignIO.read(Third_Alignment, 'clustal'):
    A.append(SeqRecord.seq)
# We will create a matrix using the numpy package.
profile = np.array(A)
difference = True
Starting_Gaps = True
# Using for loop we will go over each column in the matrix and give it 4 conditions.
for x in range(len(A[0])):
    ...
    if '-' in profile[:, x] and Starting_Gaps:
        continue
    ...
    if len(set(profile[:, x])) == 1:
        Starting_Gaps = False
        difference = False
    ...
    if len(set(profile[:, x])) != 1:...
    # Condition 4: the script will only get the index and print it if the difference is false.
```

The Full Code part 2

```
    if not difference:
        ...
        if len(y) == 1:
            Dissimilar_Positions_open.write('There is a dissimilar region at index %s \n' % y[0])
            ...
            for SeqRecord in AlignIO.read(Third_Alignment, 'clustal'):
                if SeqRecord[y[0]] != A[0][y[0]]:
                    Dissimilar_Positions_open.write(
                        'Which is %s in %s and in the Main consensus  is %s \n' %
                        (SeqRecord[y[0]], SeqRecord.id, A[0][y[0]]))
            ...
            y.clear()
        ...
        if len(y) == 0:
            continue
        ...
        if len(y) != 1:
            Dissimilar_Positions_open.write(
                'There is a dissimilar region start at index %s and end at index %s \n' % (y[0], y[-1]))
            for SeqRecord in AlignIO.read(Third_Alignment, 'clustal'):
                # While using a range of indexes the last index is not included
                # so we added 1 to the last value to include it.
                if SeqRecord.seq[y[0]:(y[-1] + 1)] != A[0][y[0]:(y[-1] + 1)]:
                    Dissimilar_Positions_open.write('Which is %s in %s and in the Main consensus  is %s \n' %
                                                    (SeqRecord.seq[y[0]:(y[-1] + 1)], SeqRecord.id,
                                                     A[0][y[0]:(y[-1] + 1)]))
            y.clear()
# Then we close the file since we have all the data we need.
Dissimilar_Positions_open.close()
```

Script Demo

- For sake of time we will use these two example lists from NCBI
- Africa : MT951858.1,MT951862.1,MT951856.1,MT951859.1,MT951860.1,MT951863.1, MT951868.1,MT951870.1,MT951876.1,MT951846.1
- Asia : MZ767421.1,MZ767468.1,MZ767470.1,MZ767495.1,MZ767545.1,MZ767437.1, MZ767481.1,MZ767508.1,MZ767514.1,MZ767499.1

Conclusion

- This study was performed on the same virus, HIV-1 but it shows that the environment has an effect in the virus development as by changing the continent the whole genome of the virus got different nucleotides which may represent different functional products at the end as shown in result. The more critical this functional products to the virus, the more the notable effect on its behavior and transmit.

Thank You