README - “Python pipeline development for protein sequence analysis”.

@AUTHOR: JOEL MOREIRA
@DATE: 04/04/2022 - 27/06/2022
@LOCATION: The Aktinson Lab
@SUPERVISOR: Gemma Aktinson

This readme file contains all the scripts and commands used during the research project on "Python pipeline development for protein sequence analysis".

Tools used: Bash, Python, MMSeqs2 and Cytoscape.
Input data: ABCF family (an in-house Atkinson lab dataset)

1) Organizing the Input data

In order to understand and organize the input data (firmicutes_table_1e-70-80_with_seq_v7.csv) the following commands were applied:

Firstly to make a list with all the evalues of the originals so it can be considered later for the evalue used on the cluster.

```
cat firmicutes_table_1e-70-80_with_seq_v7.csv |cut -d ";" -f5| sort > evaluesfirmicutes.tx
t
```

Then a list with how many sequences in each subfamily so it can be compared in the end to our final result and check the accuracy of our method.

```
cat firmicutes_table_1e-70-80_with_seq_v7.csv | cut -f 3| tail -n +2| sort -n| uniq -c > s
ubfamilies.txt
```

And finally a python script developed by me was used to make a fasta file from the original csv.

```
./csv_to_fasta.py firmicutes_table_1e-70-80_with_seq_v7.csv firmicutes.fa
```

csv_to_fasta.py:

```
#!/usr/bin/env python
'''
************ Bioinformatics: Research Project ************
***  Small python script to convert the csv file with the data into a FASTA  ***
Author(s): Joel Moreira
Description: A parser that reads the csv  file with the input dataand outputs a fasta file
with the header (containing the gcf/acc/specie/strain) and the sequence
Usage: ./csv_to_fasta.py [input] [output]
Example: ./csv_to_fasta.py firmicutes_table_1e-70-80_with_seq_v7.csv firmicutes.fa

'''

import csv
import sys
inputPath = sys.argv[1]
outputPath = sys.argv[2]

print(
"\n"
"############   The header will contain the gcf, acc, specie and strain of each seq   #####
######\n"
"...\n"
"...\n"
"..."
)

n = 18

with open(inputPath, 'r') as csvFile:
reader = csv.reader(csvFile, delimiter = '\t')
next(reader)
with open(outputPath, 'w') as outfas:
for row in reader:
n += 1
gcf = row[0]
acc =row[1]
specie = row[17]
specie= specie.replace(" ","-")
strain = row [18]
strain= strain.replace(" ","-")
dnaseq = row[8]
outfas.write('>' + gcf + '' + acc + ''+ specie + '_' + strain + '\n' + dnaseq + '\n')

print(
"############     Task Done Sucessfully   ############\n")
```

2)Installation of the MMSeqs2

```
conda create -n cyto -c conda-forge -c bioconda python=3.9 cytoscape
```

For the learning and comprehension of the MMSeqs2 software the "MMseqs2 User Guide" was used and the commands describe there were used in the beginning of the project with just a portion of the data in order to get familiar to the software.

Cytoscape was installed and ran on my personal computer and the usage requires a visual interface so no commands were utilized.

For the running of MMSeqs2:

Create a database in order to run:

```
mmseqs createdb firmicutes.fa DB
```

Run as linclust algorithm:

```
mmseqs linclust DB DB_half tmp -c 0.5 --cov-mode 0 --min-seq-id 0.5 -e 4.1e-81 > DB_half_o
utput.txt
```

Run as cluster algorithm:

```
mmseqs cluster DB DB_half tmp -c 0.5 --cov-mode 0 --min-seq-id 0.5 -e 4.1e-81 > DB_half_ou
tput.txt
```

Create a tsv file with the results so it can be open by Cytoscape and/or analyzed.

```
mmseqs createtsv DB DB DB_half DB_half.tsv
```

Run as cluster algorithm but as only as single step clustering:

```
mmseqs cluster clu3DB clu4 tmp4 -c 0.5 --single-step-clustering --cov-mode 1 --min-seq-id
 0.5 -e 4.1e-40
```

In the initial phase both linclust mode and cluster mode were used as different cover-modes (0,1 and 2), different e-values and different minimum sequence identity (0; 0.5;1).

It is also possible to run MMSeqs2 without creating a database that also creates automatically the tsv file as one of the outputs:

```
mmseqs easy-cluster ../firmicutes.fa DB_1cl tmp -c 0.5 --cluster-mode 1 --cov-mode 1 --min
-seq-id 0.5 -e 1 > 1cl_output.txt
```

During the process it was also tested clustering by phases where you create subgroups and try to cluster it even more so we would get to a better final result, here is an example doing it 2 times but it was tested with a maximum of 10 runs:

```
mmseqs cluster DB clu1 tmp21 -c 0.5 --cluster-mode 1 --cov-mode 1 --min-seq-id 0.5 -e 4.1e
-40
```

```
mmseqs createsubdb clu1 DB cluDB
```

```
mmseqs cluster cluDB clu2 tmp22 -c 0.5 --cluster-mode 1 --cov-mode 1 --min-seq-id 0.5 -e
 4.1e-40
```

```
mmseqs mergeclusters DB final_clu clu1 clu2 #Merging the clusters
```

```
mmseqs createtsv DB DB final_clu final_2_cluster.tsv #Creating the tsv file
```

3) Analyzing the results

Create a file with just the first column, organized and sorted, of the output so it can be used later:

```
cat DB_cl3f_cluster.tsv| cut -f1| uniq -c| sort -n > cl3f_clusters.txt
```

Runned a command similar to this with a python script also developed by me called mmseqs_clusters.py:

```
./mmseqs_clusters.py ./approach_1/DB_cl3f_cluster.tsv firmicutes_table_1e-70-80_with_seq_v
7.csv cl3f_clusters.txt
```

```
mmseqs_clusters.py script:

#!/usr/bin/python3


###                      ************ Bioinformatics Master ************
###                           *** Parser for tsv cluster file ***

# Date: 2022-05-12
# Author(s): Joel Moreira

# Description:
#   A parser that reads the output of mmseqs and a file created by: cat [mmseqs output]| c
ut -f1| uniq -c| sort -n > output.txt with the original input data (firmicutes table) and
 outputs the curated subfamilies and in how many different cluster they are with mmseqs.

# Usage:
# ./mmseqs_clusters.py [mmseqs output] firmicutes_table_1e-70-80_with_seq_v7.csv [file cre
ated by: cat [mmseqs output]| cut -f1| uniq -c| sort -n > output.txt]

# Usage Example:
# ./mmseqs_clusters.py ./approach_1/DB_cl3f_cluster.tsv firmicutes_table_1e-70-80_with_seq
_v7.csv cl3f_clusters.txt

# tsv_cluster_parser.py being this code
# DB_cl3f_cluster.tsv the output from mmseqs easy-clust/linclust
# firmicutes_table_1e-70-80_with_seq_v7.csv the original input data
# cl3f_clusters.txt a file created by cat [mmseqs output]| cut -f1| uniq -c| sort -n > cl3
f_clusters.txt

#%%

import sys
import re

inFile = sys.argv[1]
infile= sys.argv[2]
newFile=sys.argv[3]
#outfile=sys.argv[3]
```

```
list=[]
lets_go={}
count={}
how={}
Total=0
with open(inFile, 'r',  encoding='cp1252') as text:
    for line in text:
        first=line.rsplit("\t")[0]
        if first not in list:
            list.append(first)
        else:
            pass
    for item in list: #This list have 361 items
        a1=item.split("_",4)[0:2]
        b1=item.split("_")[2:4]
        a='_'.join(a1)
        b='_'.join(b1)
#          lets_go[a+"_"+b]="None"
        #It creates the 361 entries [len(lets_go) its 361]
        with open(infile, 'r',  encoding='cp1252') as doc2:
            for lina in doc2:
        #Here only checks the whole document once
                if a and b in lina:
                    lets_go[a+"_"+b]=lina.rsplit("\t")[2]
#with open(outfile, "w") as out1:
for key, value in lets_go.items():
#    print(key, ' -> ', value)
    with open(newFile, 'r',  encoding='cp1252') as i3:
        for lino in i3:
            if key in lino:
                if value in count:
                    count[value]+=int(lino.split()[0])
                    how[value]+=1

                else:
                    count[value]=int(lino.split()[0])
                    how[value]=1
               # print (value + ":" + lino.split()[0])
for key, value in count.items():
        print(key,":", value," in ",how[key],"clusters")
        Total+=how[key]

print("Total Clusters:",Total)
```