Multiple Sequence Alignment (MSA) di sequenze SARS-CoV-2

Silva Edoardo 816560
Marchetti Davide 815990

A.A.: 2019/2020

Abstract

L'obiettivo del progetto consiste nell'analizzare, allineare ed identificare le differenze con la sequenza di riferimento prelevata su un campione di Wuhan, progettando un formato di output nel quale memorizzare i risultati ottenuti.

Usando i sequenziamente genomici del virus denominato Covid-19, reperibili tramite NCBI[1] e GISAID[2].

Sfruttando gli strumenti messi a disposizione dall'European Bioinformatics Institute[3] abbiamo effettuato l'allineamento di un insieme di sequenze relative a paesi del medioriente.

[1] `https://www.ncbi.nlm.nih.gov/genbank/sars-cov-2-seqs/`
[2] `https://www.gisaid.org/`
[3] `https://www.ebi.ac.uk/Tools/msa/`

1 Descrizione

Il SARS-CoV-2 (acronimo dall'inglese Severe Acute Respiratory Syndrome - CoronaVirus - 2), è un ceppo virale della specie SARS-related coronavirus/SARS-CoV, facente parte del genere Betacoronavirus (ceppo di virus a RNA).

Il virus è stato sequenziato genomicamente dopo un test di acido nucleico effettuato su un campione prelevato da un paziente colpito da una polmonite, di cui non si conosceva la causa, ad inizio Dicembre 2019 a Wuhan (città continentale a est della Cina).

2 Sequenze Analizzate

In aggiunta alla sequenza di riferimento di Wuhan, sono state selezionate alcune sequenze relative a paesi dell'area mediorientale.

Reference di Wuhan

- `NC_045512.2` pubblicata il 17/01/2020 (ultimo aggiornamento)

Iran

- `MT320891.2` pubblicata il 10/04/2020
- `MT281530.2` pubblicata il 04/04/2020
- `EPI_ISL_442523` sequenziata il 09/03/2020
- `EPI_ISL_437512` sequenziata il 26/03/2020

Israele

- `MT276598.1` sequenziata il 02/04/2020
- `MT276597.1` sequenziata il 02/04/2020
- `EPI_ISL_447469` sequenziata il 14/04/2020

Pakistan

- `MT262993.1` pubblicata il 25/03/2020
- `MT240479.1` pubblicata il 25/03/2020
- `EPI_ISL_417444` sequenziata il 04/03/2020

Turchia

- `MT327745.1` pubblicata il 13/04/2020,
- `EPI_ISL_437334` sequenziata il 24/03/2020
- `EPI_ISL_437317` sequenziata il 27/03/2020

3 Strumenti utilizzati

L'analisi è stata effettuata utilizzando `Clustal Omega` e `MUSCLE`, due strumenti per l'allineamento di sequenze multiple (MSA) accessibili attraverso un'interfaccia web[4] [5].

4 Analisi preliminare

Una prima analisi delle sequenze è stata effettuata con l'ausilio di Jalview[6], un software open source che offre la possibilità di generare una visualizzazione grafica degli gli allineamenti effettuati dai tool.

Tramite questa si riescono a mettere in evidenza aspetti interessanti: come osservabile nelle fig. 1 e 2, la maggior parte delle differenze di allineamento si concentrano agli estremi delle sequenze stesse.

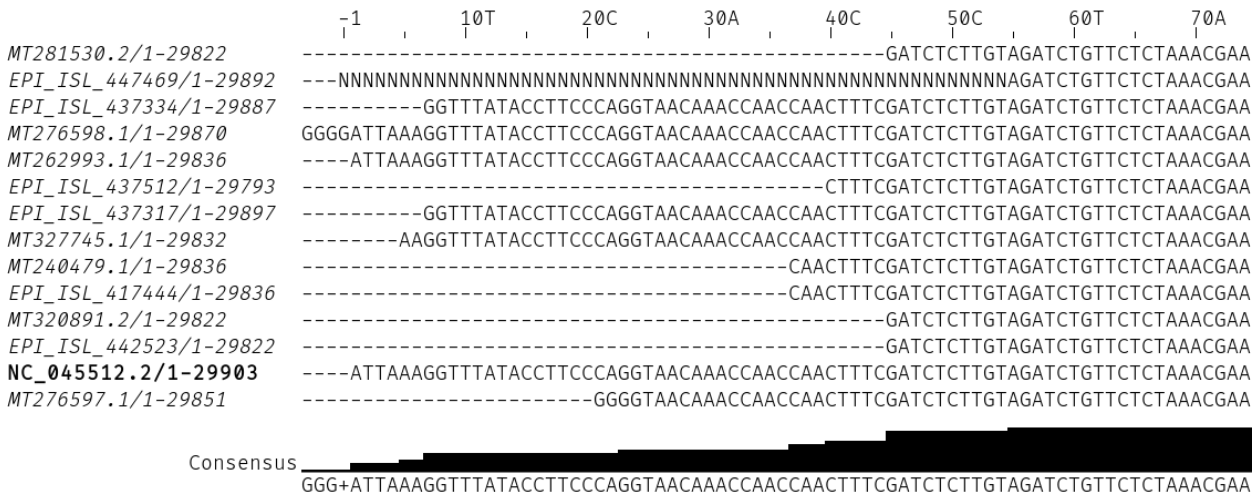

Figura 1: Differenze all'inizio dell'allineamento

[4] `https://www.ebi.ac.uk/Tools/msa/clustalo/`
[5] `https://www.ebi.ac.uk/Tools/msa/muscle/`
[6] `https://www.jalview.org/`

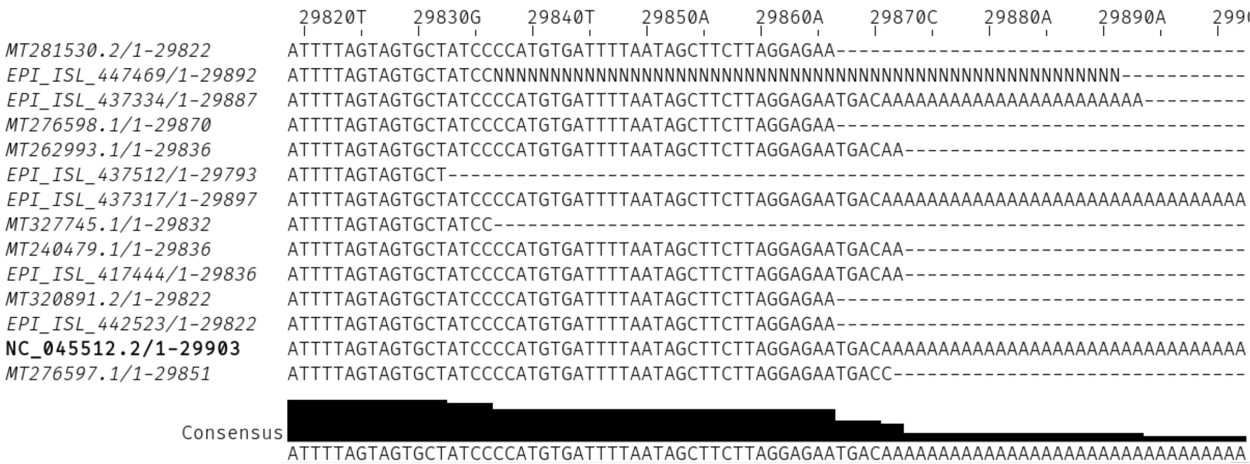

Figura 2: Differenze alla fine dell'allineamento

Inoltre, due delle sequenze selezionate presentano basi mancanti o possibili problemi nel sequenziamento (fig. 3). Quest'ultimi verranno successivamente interpretate facendo riferimento allo IUPAC Code[7] e, specificatamente, verranno considerate come match rispetto al reference.

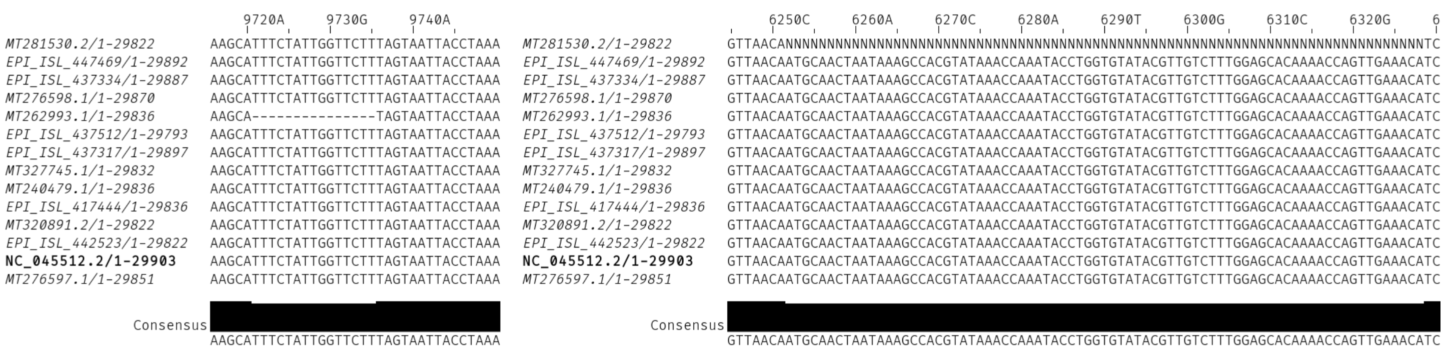

Figura 3: Differenze all'interno delle sequenze

5 Codice

Il codice è in python ed esegue le seguenti funzioni:

1. pulisce la cartella di output per far spazio ai nuovi file, usando la libreria os.

[7] `https://www.bioinformatics.org/sms/iupac.html`

2. per ogni coppia di file in input contenendo l'analisi di allineamento eseguita usando i due tool:
   (a) salva il nome del file json in ooutput seguendo l'analisi.
   (b) ne compara le differenze per verificare se esistono differenze tra i 2 tools.

5.1 librerie

- **re:** usato in **parsers.py** per dividere ogni linea tra: [id_sequenza, sequenza, posizioni] nelle funzioni di parsing.
- **hashlib:** usato in **utils.py** per creare l'hash da inserire come nome ai file JSON di output.
- **json:** usato in **utils.py** per elaborare ulteriormente l'uotput al fine di comparare i 2 tools di allineamento.
- **datetime:** usato in **utils.py** per prendere il tempo da inserire nell'hash, al fine di rendere unico l'output ed evitare sovrascritture.
- **os:** usato in **utils.py, main.py** per gestire input e output. Anche per pulire la cartella input nel main.

5.2 descrizione metodi

- `runClustal(inputFile, reference_id, nseq=3)`: funzione che esegue il parsing del file di allineamento clustal (inputFile), esegue il parsing degli allineamenti e li salva nel file di output.
  nseq serve al parser in quanto la classe `ClustalParser` richiede il numero di sequenze da elaborare nelle sue funzioni.
- `runMuscle(inputFile, reference_id, nseq=3)`: funzione che esegue il parsing del file di allineamento muscle (inputFile), esegue il parsing degli allineamenti e li salva nel file di output.
- `parse(self, filename, reference=None, list=[]`: legge il file in ingresso e restituisce il file in input suddiviso in: reference, sequenze, lunghezza sequenze. svolge la sua funzione sfruttando il metodo: `parseLines(self, lines)`
- `save(alignment, analyzer, reference_id=None, tool=None, path =None)`: crea file json di output chiamato reference_id_hash-sha1

del tempo di inzio esecuzione concatenato con sequences_ids delle sequenze.

- `jsonComp(file1, file2)`: funzione cre prende i 2 file json generati dall'elaborazione clustal e muscle e ritorna le differenze tra i 2 oggetti al fine di compararli.
- `saveCompareFile(filename="differences.txt", country="", diff =[], path='output')`: file che divide per paese prende la lista di differenze tra gli allineamenti clustal e muscle e li aggiunge al file di output "differences.txt" per mostrarli.

6 Output

Come output vengono prodotti dei file json corrispondenti al nome reference_seq_hash.hexdigest(): l'id della reference accostata all'hash sha1 con encoding utf8 dell'id delle sequenze usate concatenate al tempo di inizio lavorazione (aggiunto per evitare di sovrascrivere l'output dello stessa serie di sequenze sia con muscle che con clustal).
l'oggetto JSON è così composto:

```
{
  "reference": reference_id,
  "analyzed_sequences": id delle sequenze analizzate,
  "unmatches": {
    "(hash inizio mismatch + fine mismatch)": {
      "from": inizio mismatch,
      "to": fine mismatch,
      "sequences": sequenze contenenti la regione di mismatch
    }
  }
}
```

7 Conclusioni

La maggior parte dei disallineamenti si trovano entro le prime 100 basi e dopo la 2900esima, con pochi disallineamenti sporadici all'unterno.
Clustal tronca id troppo lunghi, i 2 tool di allineamento non eseguono allineamenti identici.
...

8 references