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

Edoardo Silva 816560
Davide Marchetti 815990

A.A.: 2019/2020

1 Abstract

La seconda parte del progetto prevede di elaborare i file prodotti in precedenza ricavando informazioni relative alle alterazioni rilevate e producendo in output una tabella riassuntiva contenente:

- il gene id del gene in cui cade la variazione con lo start e l'end della sua CDS rispetto alla reference
- il codone (o i codoni) alterato della reference, con posizione di inizio rispetto alla CDS, sequenza del codone e amminoacido codifcato
- il nuovo codone generato dalla variazione (o i nuovi codoni generati) specifcando la sequenza del codone e il nuovo amminoacido codifcato

2 Algoritmo

L'algoritmo inizia caricando tutti i file necessari per l'elaborazione, in particolare quelli prodotti in output nella parte precedente del progetto:

1. Caricamento della sequenza reference dal file corrispondente memorizzato in `/project-1/input/reference.fasta`.

2. Caricamento di uno dei file di output prodotti nella prima parte di progetto. Nel nostro caso è stato utilizzata l'analisi dell'allineamento di `ClustalW`.

3. Lettura del file `Genes-CDS.xlsx` contenente le informazioni sui geni e le CDS della sequenza di reference. In particolare, per le CDS che derivano dalla join di due sequenze è possibile specificare il punto di unione della sequenza.

Dopo la lettura del materiale rilevante a questa fase di elaborazione, l'algoritmo itera le variazioni rilevate nell'allineamento e per ciascuna di esse esegue i seguenti step:

1. Trova le CDS nelle quali avviene l'alterazione rispetto alla reference.

2. Recupera le informazioni del gene associato alle CDS rilevate calcolando le posizioni globali e relative alla CDS dell'alterazione.

3. Identifica i codoni alterati e ne effettua la ritraduzione in amminoacidi grazie ad una look-up table (listato 1). Vengono ignorate le alterazioni che presentano sequenze di soli `-`, derivate probabilmente da un sequenziamento errato o un'alterazione posta ai capi dell'allineamento.

4. Memorizza tutte le informazioni ricavate in una struttura dati tramite cui derivare la tabella per l'output finale associando i valori a chiavi prestabilite (listato 2).

Al termine dell'elaborazione di tutte le alterazioni, viene costruito un oggetto di tipo `DataFrame` fornito dalla libreria `pandas`.

Le chiavi utilizzate nella costuzione della struttura dati a lista diventeranno le colonne del `DataFrame`. Questo sarà esportato in CSV nella cartella `/project-2/output/alteration-table.csv` per permettere una visualizzazione più semplice tramite programmi terzi.

3 Informazioni memorizzate

Ad ogni variazione analizzata corrisponde un entrata nella struttura dati a lista contenente le seguenti informazioni:

- **gene_id**: id del gene in cui cade la variazione
- **gene_start**: inizio del gene in cui cade la variazione (1-based)
- **gene_end**: fine del gene in cui cade la variazione (1-based)
- **cds_start**: inizio della `Coding DNA Sequence` della porzione del gene in cui cade la variazione (1-based)
- **cds_end**: fine della `Coding DNA Sequence` della porzione del gene in cui cade la variazione (1-based)
- **original_codone**: codone della reference prima della modifica
- **altered_codone**: codone della reference modificati dalla variazione
- **relative_start**: inizio della variazione in rispetto all'inizio della cds (1-based)
- **relative_end**: fine della variazione in rispetto all'inizio della cds (1-based)
- **alteration**: sequenza della variazione
- **encoded_aminoacid**: amminoacido codificato da **altered_codone**

4 Output

TODO Screenshot della tabella

5 Listati di codice

Code Listing 1: Tabella per la traduzione in amminoacidi

```
aminoacids_lookup_table = {
    'START': 'ATG',
    'STOP': ['TAA', 'TAG', 'TGA'],
    'F': ['TTT', 'TTC'],
    'L': ['TTA', 'TTG', 'CTT', 'CTA', 'CTC', 'CTG'],
    'I': ['ATT', 'ATC', 'ATA'],
    'M': ['ATG'],
    'V': ['GTT', 'GTA', 'GTC', 'GTG'],
    'S': ['TCT', 'TCA', 'TCC', 'TCG', 'AGT', 'AGC'],
    'P': ['CCT', 'CCA', 'CCC', 'CCG'],
    'T': ['ACT', 'ACA', 'ACC', 'ACG'],
    'A': ['GCT', 'GCA', 'GCC', 'GCG'],
    'Y': ['TAT', 'TAC'],
    'H': ['CAT', 'CAC'],
    'Q': ['CAA', 'CAG'],
    'N': ['AAT', 'AAC'],
    'K': ['AAA', 'AAG'],
    'D': ['GAT', 'GAC'],
    'E': ['GAA', 'GAG'],
    'C': ['TGT', 'TGC'],
    'W': ['TGG'],
    'R': ['CGT', 'CGA', 'CGC', 'CGG', 'AGA', 'AGG'],
    'G': ['GGT', 'GGA', 'GGC', 'GGG']
}
```

Code Listing 2: Memorizzazione dei risultati nella struttura dati a lista

```
for key, value in variations:
  for index, cds in affected_cdses.iterrows():
    ...
    variations_to_genes.append({
      'gene_id': gene_id,
      'gene_start': gene_start + 1, # 1-based position
      'gene_end': gene_end,
      'cds_start': cds_start + 1, # 1-based position
      'cds_end': cds_end,
      'original_codone': original_codone,
      'altered_codone': altered_codone,
      'relative_start': relative_start + 1, # 1-based position
      'relative_end': relative_end,
      'alteration': sequence,
      'original_aminoacid': original_aminoacid,
      'encoded_aminoacid': encoded_aminoacid
    })
```