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

Edoardo Silva 816560
Davide Marchetti 815990

A.A.: 2019/2020

## 1 Abstract

Il lavoro svolto è diviso in 2 parti e il codice in 4 sezioni:

### 1.1 Matrice delle variazioni

Qui si trovano le prime 2 parti del codice: il recupero per intero delle 14 sequenze usate e la lettura dei file di variazione generati in output nella parte1

**Listing 1:** Porzione di ciclo

```
reference_id = load_fasta_id(os.path.join('..', '..', 'project
    ↪ -1', 'input', 'reference.fasta'))
sequence_ids = read_sequence_ids(paths=[
    os.path.join('..', '..', 'project-1', 'input', 'GISAID'),
    os.path.join('..', '..', 'project-1', 'input', 'ncbi'),
])
sequence_ids.insert(0, reference_id) #insert reference no
    ↪ variations

...

clustal_output = load_output('Clustal-NC_045512.2_2020-05-30_16
    ↪ -51.json')
variations = clustal_output['unmatches'].items()
```

Infine nella parte 3 del nostro codice generiamo la tabella delle variazioni **'table.csv'** per variazione, per poi trasporla.

**Listing 2:** Porzione di ciclo

```
...

for key, value in variations:
       row = np.zeros(len(sequence_ids))
       indexes.append('C{}'.format(counter))
       for sequence in value['sequences']:
           row[sequence_ids.index(sequence)] = 1
       rows.append(row)
       counter += 1

...

trait_matrix = pd.DataFrame(rows, index=indexes, columns=
    ↪ sequence_ids, dtype=bool).transpose()
trait_matrix = phylogeny.reorder_columns(trait_matrix, axis=0,
    ↪ ascending=False)
trait_matrix.to_csv(os.path.join('..', 'output', 'table.csv'))
```

## 1.2 Albero filogenetico

La quarta e ultima parte del codice consiste nel ricostrure l'albero filogenetico prendendo l'output del punto 3 e facendo estensivo uso di funzioni create per questo lavoro nel file **'phylogeny.py'** per:

1. generare matrice compatibile con filogenesi perfetta.
2. assicurarsi che tutto abbia funzionato e non sia più presente matrice proibita.
3. generare e stampare in output l'albero delle sequenze.

**Listing 3:** Porzione di ciclo

```
candidate_matrix = get_perfect_phylogeny_character_matrix(
    ↪ trait_matrix)
if phylogeny.is_forbidden_matrix(candidate_matrix):
       raise Exception('Invalid perfect phylogeny matrix')
phylogeny.build_tree(candidate_matrix)
```

La generazione dell'albero viene svolta come segue:

1. Riordino il dataframe ordinando le colonne per numero decrescende di 1

**Listing 4:** Porzione di ciclo

```
    sorted_axis = df.sum(axis=0).sort_values(ascending=False)
    if axis == 0:
        return df[sorted_axis.index]
    return df.reindex(sorted_axis.index)
```

2. Genera nodo radice, poi per ogni sequenza genera nodi da collegare a root se incontra la presenza di variazione per ogni nodo non già presente

**Listing 5:** Porzione di ciclo

```
for i, row in df.iterrows():
        current_node = root

        for j in range(len(row)):
            # If alteration is present in the current sequence
            if row.iloc[j]: #true in cell
                key = j
                # If current_node is already linked to the j-th
                    ↪ variation
                # edges = dictionary with links from (
                    ↪ current_node, u)
                if key in current_node.edges:
                    # Update the current node
                    current_node = current_node.edges[key]
                else:
                    # Create a new node u and link it with the
                        ↪ last node of this sequence
                    u = Node('U-{}'.format(row.index[j]), edges
                        ↪ ={}, parent=current_node)
                    current_node.edges[key] = u
                    current_node = u
        # We looped all the variations, insert the sequence node
            ↪  at the end of the chain
        Node(i, parent=current_node)
```

3. Conversione dell'albero precedentemente generato in un newick_tree elaborabile dalla libreria **Phylo**, usata per la visualizzazione.

**Listing 6:** Porzione di ciclo

```
newick_tree = to_newick_tree(root, intermediates=False)
newick_tree = Phylo.read(io.StringIO(newick_tree), 'newick')
```

4. Generazione immagine output

**Listing 7:** Porzione di ciclo

```
fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(1, 1, 1)

    # Assign sequences data to the leaf
    root = newick_tree.clade
    merge_sequences_data(root, sequences_data) # Add sequence
        ↪ data to sequence leaves
    # Render with plt
    Phylo.draw(newick_tree, do_show=False, axes=ax)

    ax.set_xlabel('Number of alterations')
    ax.set_ylabel('Sequences')
    plt.tight_layout()
    plt.savefig(os.path.join('..', 'output', 'phylogenetic-tree
        ↪ .png'))
    plt.show()
```

## 2 Confronto alberi

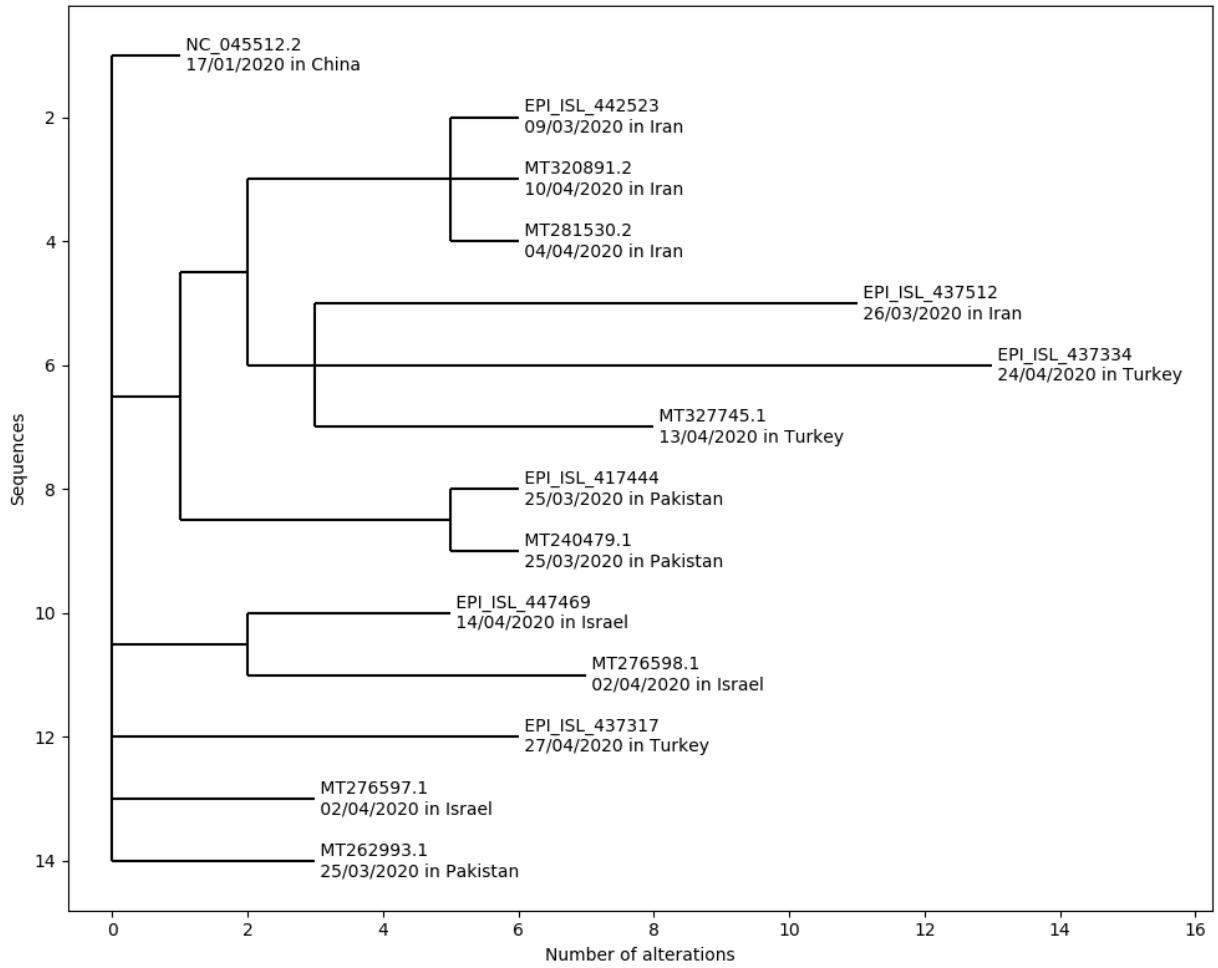


**Figura 1:** Albero di output dello script

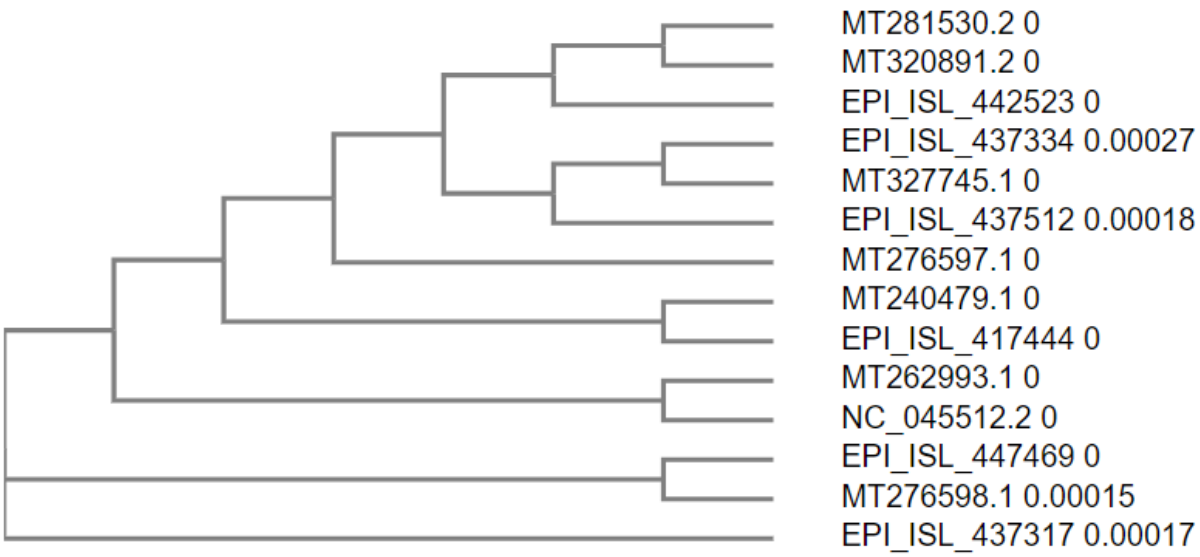


**Figura 2:** Albero di output del sito delle sequenze

## 3 conclusione