

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

EDOARDO SILVA 816560  
DAVIDE MARCHETTI 815990

A.A.: 2019/2020

## 1 Abstract

Per ogni variazione registrata nell'output della parte1, viene creata un'entità nella tabella di output contenente:

- **gene\_id**: id del gene in cui cade la variazione
- **gene\_start**: inizio del gene in cui cade la variazione
- **gene\_end**: fine del gene in cui cade la variazione
- **cds\_start**: inizio della *coding DNA sequence* della porzione del gene in cui cade la variazione
- **cds\_end**: fine della *coding DNA sequence* della porzione del gene in cui cade la variazione
- **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
- **relative\_end**: fine della variazione in rispetto all'inizio della cds
- **sequence**: variazione
- **encoded\_aminoacid**: amminoacido codificato da **altered\_codone**

## 2 codice

Il codice inizia caricando la sequenza di reference dal file **'reference.fasta'** e un file di output.

Infine carica in una lista le CDS della reference dal file **'Genes-CDS.xlsx'**.

Il corpo del codice esegue un ciclo per ogni variazione nel file di output

**Listing 1:** Porzione di ciclo

```
{
for key, value in variations:
    ...

    affected_cds = df_ref_cds.loc[(value['from'] > df_ref_cds['
        ↪ from']) & (value['to'] < df_ref_cds['to'])]

    if not affected_cds.empty:
        gene = df_ref_genes.loc[affected_cds['GeneID']]
        gene_id = affected_cds.iloc[0]['GeneID']
        gene_start = gene.iloc[0]['Start']
        gene_end = gene.iloc[0]['End']
        cds_start = affected_cds.iloc[0]['from']
        cds_end = affected_cds.iloc[0]['to']
        sequence = value['alt']
        relative_start = value['from'] - gene_start
        relative_end = relative_start + len(sequence),

    ...
}
```

e riempie la tabella

**Listing 2:** Appending nel file di output

```
{
variations_to_genes.append({
    'gene_id': gene_id,
    'gene_start': gene_start,
    'gene_end': gene_end,
    'cds_start': cds_start,
    'original_codone': original_codone,
    'altered_codone': altered_codone,
    'relative_start': relative_start + 1,
    'relative_end': relative_end,
    'sequence': sequence,
    'original_aminoacid': original_aminoacid,
```

```
        'encoded_aminoacid': encoded_aminoacid
    })
}
```

da generare come file output.

**Listing 3:** 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']
}
}
```

### 3 output

**Listing 4:** Generazione di output

```
columns = ['gene_id', 'gene_start', 'gene_end', 'cds_start',
  ↳ 'cds_end', 'original_codone', 'altered_codone', '
  ↳ relative_start', 'relative_end', 'sequence', '
  ↳ original_aminoacid', 'encoded_aminoacid']
df_variations_to_genes = pd.DataFrame(variations_to_genes,
  ↳ columns=columns)
df_variations_to_genes.to_csv(os.path.join '..', 'output', '
  ↳ out.csv'))
```

Genera file **out.csv** nella sottocartella **output** :

## 4 conclusione

Eccetto per la variazione con cancellazione di sequenze, tutti gli altri codoni modificati sono traducibili.