Priya Narain

Anna Ritz

Biology 131

14 May 2020

Project Report

I performed alignment on COVID-19(coronavirus) and MERS-CoV(Middle Eastern Respiratory Syndrome) ORF1a polyprotein sequences. MERS belongs to the betacoronavirus category, a subtype of the coronavirus family, and SARS-CoV-2 is the novel coronavirus that led to the upper-respiratory disease commonly known as COVID. Sequences from the NIH website for the ORF1a polyprotein were available, which allowed for the comparisons between the two viruses. Performing alignments on the two strings allows us to see the similarities and differences in the viral and genomic nature of the diseases. I defined the longest common subsequence and local alignment functions, and had two files for the COVID and MERS polyprotein sequence FASTA files.

I modified code from homework #10 on sequence alignment and used the longest common subsequence in order to find out the longest sequence commonalities that exist between the two strings. LCS identifies commonalities in the sequences that appear in the same relative order. When running the LCS function, the output I get shows that there are no common subsequences.

```
def LCS(string1,string2):
  # Finds long sequence commonalities between the two strings
  table = initializeTable(len(string1)+1,len(string2)+1)
  backtrack = initializeTable(len(string1)+1,len(string2)+1)

  for i in range(len(table)):
    for j in range(len(table[i])):
      if i==0 and j==0:
        table[i][j] = 0
        backtrack[i][j] = '*'
      if i>0 and j==0:
        table[i][j] = table[i-1][j]
        backtrack[i][j] = 'a'
      if i==0 and j>0:
        table[i][j] = table[i][j-1]
        backtrack[i][j] = 'b'
      if i>0 and j>0:
        m = 0
        if string1[i-1]==string2[j-1]:
          m = 1
        table[i][j] = max([table[i-1][j],table[i][j-1],table[i-1]
        [j-1]+m])
        if table[i][j] == table[i-1][j]:
          backtrack[i][j] = 'a'
        elif table[i][j] == table[i][j-1]:
          backtrack[i][j] = 'b'
        else:
          backtrack[i][j] = 'c'
```

```
  align1 = ''
  align2 = ''
  i = len(string1)
  j = len(string2)
  while i>0 or j>0:
    if backtrack[i][j] == 'c':
      align1 = string1[i-1]+align1
      align2 = string2[j-1]+align2
      i = i-1
      j = j-1
    elif backtrack[i][j] == 'b':
      align1 = '-'+align1
      align2 = string1[j-1]+align2
      j = j-1
    else:
      align1 = string2[i-1]+align1
      align2 = '-'+align2
      i = i-1

  lcs_length = table[len(string1)][len(string2)]
  return lcs_length,align1,align2
```

When running the LCS function, the output I get shows that there are no long common subsequences.

Next I attempted to perform local alignment on the two sequences. I think I had issues with this function because my strings were really long and probably put a strain on repl.it. I

really wanted this function to work out so I could arrange the two sequences to find areas of similarity, but unfortunately my output does show any results for the local alignment. Performing alignments using toy-alignments would probably have been more efficient through repl.it, and my strings were extensively long. If I were to perform this function again using long strings, I would not run it in repl.it.

```
def localAlign(string1,string2,indel,match,mismatch):
  table = initializeTable(len(string1)+1,len(string2)+1)
  backtrack = initializeTable(len(string1)+1,len(string2)+1)
  for i in range(len(table)):
    for j in range(len(table[i])):
      if i==0 and j==0:
        table[i][j] = 0
        backtrack[i][j] = '*'
      if i>0 and j==0:
        table[i][j] = table[i-1][j]+indel
        backtrack[i][j] = 'a'
      if i==0 and j>0:
        table[i][j] = table[i][j-1]+indel
        backtrack[i][j] = 'b'
      if i>0 and j>0:
        m = mismatch
        if string1[i-1]==string2[j-1]:
          m = match
        table[i][j] = max([table[i-1][j]+indel,table[i][j-1]
        +indel,table[i-1][j-1]+m])
        if table[i][j] == table[i-1][j]+indel:
          backtrack[i][j] = 'a'
        elif table[i][j] == table[i][j-1]+indel:
          backtrack[i][j] = 'b'
        else:
          backtrack[i][j] = 'c'
      if table[i][j]<0:
        table[i][j] = 0
        backtrack[i][j] = '*'
```

```
max_val = 0
i = 0
j = 0
for this_i in range(len(table)):
  for this_j in range(len(table[this_i])):
    if table[this_i][this_j] > max_val:
     max_val = table[this_i][this_j]
     i = this_i
     j = this_j
score = table[i][j]

# backtrack
align1 = ''
align2 = ''
while backtrack[i][j] != '*':
  print(backtrack[i][j])
  if backtrack[i][j] == 'c':
    align1 = string1[i-1]+align1
    align2 = string2[j-1]+align2
    i = i-1
    j = j-1
  elif backtrack[i][j] == 'b':
    align1 = '-'+align1
    align2 = string2[j-1]+align2
    j = j-1
  else:
    align1 = string1[i-1]+align1
    align2 = '-'+align2
    i = i-1
```

Ultimately, I would have liked for my local alignment function to have worked. I concluded that though the two viruses belong to the coronavirus category, there are obvious differences and the fact that there were no commonalities found with LCS shows that the sequences I worked with are very different. Sequencing the entire genome would be much more time consuming, but would probably be more efficient at detecting differences in the viral and genomic nature of the diseases.

I am okay with my report being posted online.

Link to the repl:

https://repl.it/@prinarain/Bio-131-Final

FASTA files from the NIH website:

Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome:

https://www.ncbi.nlm.nih.gov/nuccore/NC_045512.2?report=fasta&from=266&to=21555

Middle East respiratory syndrome coronavirus, complete genome:

https://www.ncbi.nlm.nih.gov/nuccore/NC_019843.3?report=fasta&from=279&to=21514