In Class Activity

Together we will write a program to distinguish which of a list of common flu virus strains were present in a given sample to see which virus a person has! In the 2010-2011 flu season three flu strains were present:

- A/California/07/2009
- A/Perth/16/2009
- B/Brisbane/60/2008

The data you have recieved from the lab was the following sequence:
TTTAAAGCAGGGGATAATTCTATTAACCATGAAGACTATCATTGCTTTGAGCTACATTCTATGTCTGGTTTTCGCTCAAAAACTTCCT

The three flu strains for that year were taken from the RefSeq database (https://www.ncbi.nlm.nih.gov/refseq/)

- A/California/07/2009
  ATGAAGGCAATACTAGTAGTTCTGCTATATACATTTGCAACCGCAAATGCAGACACATTATGTATAGGTTATCATGCGAACAAT
- A/Perth/16/2009
  AGCAGAAGCAGAGCATTTTCTAATATCCACAAAATGAAGGCAATAATTGTACTACTCATGGTAGTAACATCCAATGCAGATCGA
- B/Brisbane/60/2008
  AAAGCAGGGGATAATTCTATTAACCATGAAGACTATCATTGCTTTGAGCTACATTCTATGTCTGGTTTTCGCTCAAAAACTTCCT

We want to do this following:

- Analyse the original sequence
  - Remove the adapters from the sequence (first 3 and last 3 bases)
  - Identify the amount of each base
  - Identify the number of CG pairs
- Use the code from the analysis above to write a program that
  - Takes a list of input sequences
  - Loops through each sequence and runs the analysis

In [13]:

```
#declare variable
data="TTTAAAGCAGGGGATAATTCTATTAACCATGAAGACTATCATTGCTTTGAGCTACATTCTATGTCTGGTTTTCGCTCAAAAACTTCCTGGAA
AACAGCACGGCAACGCTGTGCCTTGGGCACCATGCAGTACCAAACGGAACGATAGTGAAAACAATCACGAATGACCAAATTGAAGTTACTAATGCTAC
TGGTTCAGAGTTCCTCAACAGGTGAAATATGCGACAGTCCTCATCAGATCCTTGATGGAAAAAACTGCACACTAATAGATGCTCTATTGGGAGACCCT
TGATGGCTTCCAAAATAAGAAATGGGACCTTTTTGTTGAACGCAGCAAAGCCTACAGCAACTGTTACCCTTATGATGTGCCGGATTATGCCTCCCTTA
CTAGTTGCCTCATCCGGCACACTGGAGTTTAACAATGAAAGCTTCAATTGGACTGGAGTCACTCAAAACGGAACAAGCTCTGCTTGCATAAGGAGATC
ACAGTTTCTTTAGTAGATTGAATTGGTTGACCCACTTAAACTTCAAATACCCAGCATTGAACGTGACTATGCCAAACAATGAACAATTTGACAAATTG
TTGGGGGGTTCACCACCCGGGTACGGACAAAGACCAAATCTTCCTGTATGCTCAAGCATCAGGAAGAATCACAGTCTCTACCAAAAGAAGCCAACAAA
AGCCCGAATATCGGATCTAGACCCAGAGTAAGGAATATCCCTAGCAGAATAAGCATCTATTGGACAATAGTAAAACCGGGAGACATACTTTTGATTAA
CAGGGAATCTAATTGCTCCTAGGGGTTACTTCAAAATACGAAGTGGGAAAAGCTCAATAATGAGATCAGATGCACCCATTGGCAAATGCAATTCTGAA
CACTCCAAATGGAAGCATTCCCAATGACAAACCATTCCAAAATGTAAACAGGATCACATACGGGGCCTGTCCCAGATATGTTAAGCAAAACACTCTGA
GCAACAGGGATGCGAAATGTACCAGAGAAGAAAACACCCTTGAAGCTGGAATTT"
```

In [14]:

```
#Remove adapters
#forward
dna1=data[3:]
#reverse
dna=dna1[:-3]
dna
```

Out[14]:

```
'AAAGCAGGGGATAATTCTATTAACCATGAAGACTATCATTGCTTTGAGCTACATTCTATGTCTGGTTTTCGCTCAAAAACTTCCTGGAAATGACAAC
GGCAACGCTGTGCCTTGGGCACCATGCAGTACCAAACGGAACGATAGTGAAAACAATCACGAATGACCAAATTGAAGTTACTAATGCTACTGAGCTGG
AGTTCCTCAACAGGTGAAATATGCGACAGTCCTCATCAGATCCTTGATGGAAAAAACTGCACACTAATAGATGCTCTATTGGGAGACCCTCAGTGTGA
TCCAAAATAAGAAATGGGACCTTTTTGTTGAACGCAGCAAAGCCTACAGCAACTGTTACCCTTATGATGTGCCGGATTATGCCTCCCTTAGGTCACTA
CTCATCCGGCACACTGGAGTTTAACAATGAAAGCTTCAATTGGACTGGAGTCACTCAAAACGGAACAAGCTCTGCTTGCATAAGGAGATCTAAAAACA
TTTAGTAGATTGAATTGGTTGACCCACTTAAACTTCAAATACCCAGCATTGAACGTGACTATGCCAAACAATGAACAATTTGACAAATTGTACATTTG
TTCACCACCCGGGTACGGACAAAGACCAAATCTTCCTGTATGCTCAAGCATCAGGAAGAATCACAGTCTCTACCAAAAGAAGCCAACAAACCGTAAGC
TATCGGATCTAGACCCAGAGTAAGGAATATCCCTAGCAGAATAAGCATCTATTGGACAATAGTAAAACCGGGAGACATACTTTTGATTAACAGCACAG
CTAATTGCTCCTAGGGGTTACTTCAAAATACGAAGTGGGAAAAGCTCAATAATGAGATCAGATGCACCCATTGGCAAATGCAATTCTGAATGCATCAC
ATGGAAGCATTCCCAATGACAAACCATTCCAAAATGTAAACAGGATCACATACGGGGCCTGTCCCAGATATGTTAAGCAAAACACTCTGAAATTGGCA
```

```
GATGCGAAATGTACCAGAGAAGAAAACACCCTTGAAGCTGGAA'
```

In [15]:

```
#Identify amount of each base
#A
pc_A= (dna.count("A")/len(dna))*100 # %A
#T
pc_T= (dna.count("T")/len(dna))*100
#C
pc_C= (dna.count("T")/len(dna))*100
#G
pc_G= (dna.count("G")/len(dna))*100

#print the outputs
print(pc_A)
print(pc_T)
print(pc_C)
print(pc_G)
```

```
34.14179104477612
23.32089552238806
23.32089552238806
20.42910447761194
```

In [16]:

```
#Identify number of CG
number_CG=dna.count("CG")
#print output
print(number_CG)
```

```
21
```

In [17]:

```
##loops help us run the same process on a number of similar things
#if i have a list of words and I want to print them all
words=["apple","orange","pear","strawberry"]
for i in words:
    print(i)

#this is helpful as it is efficient and saves time!
```

```
apple
orange
pear
strawberry
```

In [18]:

```
#dictionaries are like lists except each value has a name/key
test_dictionary={"name":"john","age":"11","class":"5th"}

print(test_dictionary)
#names/keys
print(test_dictionary.keys())
#values
print(test_dictionary.values())
#item i.e name and value
print(test_dictionary.items())
```

```
{'name': 'john', 'age': '11', 'class': '5th'}
dict_keys(['name', 'age', 'class'])
dict_values(['john', '11', '5th'])
dict_items([('name', 'john'), ('age', '11'), ('class', '5th')])
```

In [19]:

```
#Writing loops
#we can do the same thing for our analysis!!
#for each sequence calculate the percentages and number of CG's!!
test_dna_dictionary={"x":"ATATG","y":"AGCATGAGCG"}
for key,value in test_dna_dictionary.items():
    pc_A= (value.count("A")/len(value))*100 # %A
    pc_T= (value.count("T")/len(value))*100 # %T
    pc_G= (value.count("G")/len(value))*100 # %G
    pc_C= (value.count("C")/len(value))*100 # %C
    print ('Percentages of A, C, G and T for',key, 'are :', pc_A, pc_C, pc_G, pc_T) #print percenta
ges of each

    num_CGs= value.count("CG") #count number of cg
    print ('Total number of CGs in sequence: ', num_CGs)
```

```
Percentages of A, C, G and T for x are : 40.0 0.0 20.0 40.0
Total number of CGs in sequence:  0
Percentages of A, C, G and T for y are : 30.0 20.0 40.0 10.0
Total number of CGs in sequence:  1
```

In [20]:

```
#Taking inputs
#What if we have a lot of sequences?
#we can ask the program to ask us to input them and get it to make the dictionary for us!!!
#we will make two lists
#one with data, the other with names and combine them into a dictionary
#dna
all_dna= input("Enter Sequences to be tested:")
#tell python to split them by a space!
all_dna = all_dna.split(" ")

#names
all_names= input("Enter names of Sequences:")
#tell python to split them by a space!
all_names = all_names.split(" ")

#combine
combined = dict(zip(all_names, all_dna))


#print our dictionary
print(combined)
```

```
Enter Sequences to be tested:xxx yyy
Enter names of Sequences:x y
{'x': 'xxx', 'y': 'yyy'}
```

In [10]:

```
##Now we can put it all together!!
##Use your corrected test data and the three sequences above!
##My Flu Testing Program!
#Taking inputs
#we will input two lists
#one with data, the other with names and combine them into a dictionary
#dna
all_dna= input("Enter Sequences to be tested:")
#tell python to split them by a space!
all_dna = all_dna.split(" ")

#names
all_names= input("Enter names of Sequences:")
#tell python to split them by a space!
all_names = all_names.split(" ")

#combine
combined = dict(zip(all_names, all_dna))


#print our dictionary
print(combined)
```

```python
##loop through dictionary
for key,value in combined.items():
    pc_A= (value.count("A")/len(value))*100 # %A
    pc_T= (value.count("T")/len(value))*100 # %T
    pc_G= (value.count("G")/len(value))*100 # %G
    pc_C= (value.count("C")/len(value))*100 # %C
    print ('Percentages of A, C, G and T for',key, 'are :', pc_A, pc_C, pc_G, pc_T) #print percenta
ges of each

    num_CGs= value.count("CG") #count number of cg
    print ('Total number of CGs in sequence: ', num_CGs)
```

```
Enter Sequences to be
tested:AAAGCAGGGGATAATTCTATTAACCATGAAGACTATCATTGCTTTGAGCTACATTCTATGTCTGGTTTTCGCTCAAAAACTTCCTGGAAATG
CAGCACGGCAACGCTGTGCCTTGGGCACCATGCAGTACCAAACGGAACGATAGTGAAAACAATCACGAATGACCAAATTGAAGTTACTAATGCTACTGA
GTTCAGAGTTCCTCAACAGGTGAAATATGCGACAGTCCTCATCAGATCCTTGATGGAAAAAACTGCACACTAATAGATGCTCTATTGGGAGACCCTCAG
ATGGCTTCCAAAATAAGAAATGGGACCTTTTTGTTGAACGCAGCAAAGCCTACAGCAACTGTTACCCTTATGATGTGCCGGATTATGCCTCCCTTAGGT
AGTTGCCTCATCCGGCACACTGGAGTTTAACAATGAAAGCTTCAATTGGACTGGAGTCACTCAAAACGGAACAAGCTCTGCTTGCATAAGGAGATCTAA
AGTTTCTTTAGTAGATTGAATTGGTTGACCCACTTAAACTTCAAATACCCAGCATTGAACGTGACTATGCCAAACAATGAACAATTTGACAAATTGTAC
GGGGGGTTCACCACCCGGGTACGGACAAAGACCAAATCTTCCTGTATGCTCAAGCATCAGGAAGAATCACAGTCTCTACCAAAAGAAGCCAACAAACCG
CCCGAATATCGGATCTAGACCCAGAGTAAGGAATATCCCTAGCAGAATAAGCATCTATTGGACAATAGTAAAACCGGGAGACATACTTTTGATTAACAG
GGGAATCTAATTGCTCCTAGGGGTTACTTCAAAATACGAAGTGGGAAAAGCTCAATAATGAGATCAGATGCACCCATTGGCAAATGCAATTCTGAATGC
CTCCAAATGGAAGCATTCCCAATGACAAACCATTCCAAAATGTAAACAGGATCACATACGGGGCCTGTCCCAGATATGTTAAGCAAAACACTCTGAAAT
AACAGGGATGCGAAATGTACCAGAGAAGAAAACACCCTTGAAGCTGGAA
ATGAAGGCAATACTAGTAGTTCTGCTATATACATTTGCAACCGCAAATGCAGACACATTATGTATAGGTTATCATGCGAACAATTCAACAGACACTGTA
CAGTACTAGAAAAGAATGTAACAGTAACACACTCTGTTAACCTTCTAGAAGACAAGCATAACGGGAAACTATGCAAACTAAGAGGGGTAGCCCCATTGC
GGGTAAATGTAACATTGCTGGCTGGATCCTGGGAAATCCAGAGTGTGAATCACTCTCCACAGCAAGCTCATGGTCCTACATTGTGGAAACACCTAGTTC
AATGGAACGTGTTACCCAGGAGATTTCATCGATTATGAGGAGCTAAGAGAGCAATTGAGCTCAGTGTCATCATTTGAAAGGTTTGAGATATTCCCCAAG
GTTCATGGCCCAATCATGACTCGAACAAAGGTGTAACGGCAGCATGTCCTCATGCTGGAGCAAAAAGCTTCTACAAAAATTTAATATGGCTAGTTAAAA
AAATTCATACCCAAAGCTCAGCAAATCCTACATTAATGATAAAGGGAAAGAAGTCCTCGTGCTATGGGGCATTCACCATCCATCTACTAGTGCTGACCA
AGTCTCTATCAGAATGCAGATGCATATGTTTTTGTGGGGTCATCAAGATACAGCAAGAAGTTCAAGCCGGAAATAGCAATAAGACCCAAAGTGAGGGRT
AAGGGAGAATGAACTATTACTGGACACTAGTAGAGCCGGGAGACAAAATAACATTCGAAGCAACTGGAAATCTAGTGGTACCGAGATATGCATTCGCAA
AAGAAATGCTGGATCTGGTATTATCATTTCAGATACACCAGTCCACGATTGCAATACAACTTGTCAAACACCCAAGGGTGCTATAAACACCAGCCTCCC
CAGAATATACATCCGATCACAATTGGAAAATGTCCAAAATATGTAAAAAGCACAAAATTGAGACTGGCCACAGGATTGAGGAATATCCCGTCTATTCAA
GAGGCCTATTTGGGGCCATTGCCGGTTTCATTGAAGGGGGGTGGACAGGGATGGTAGATGGATGGTACGGTTATCACCATCAAAATGAGCAGGGGTCAG
TGCAGCCGACCTGAAGAGCACACAGAATGCCATTGACGAGATTACTAACAAAGTAAATTCTGTTATTGAAAAGATGAATACACAGTTCACAGCAGTAG
AGCAGAAGCAGAGCATTTTCTAATATCCACAAAATGAAGGCAATAATTGTACTACTCATGGTAGTAACATCCAATGCAGATCGAATCTGCACTGGGATA
CGTCAAACTCACCACATGTCGTCAAAACTGCTACTCAAGGGGAGGTCAATGTGACTGGTGTAATACCACTGACAACAACACCCACCAAATCTCATTTTG
TCTCAAAGGAACAGAAACCAGGGGGAAACTATGCCCAAAATGCCTCAACTGCACAGATCTGGACGTAGCCTTGGGCAGACCAAAATGCACGGGGAAAAT
TCGGCAAGAGTTTCAATACTCCATGAAGTCAGACCTGTTACATCTGGGTGCTTTCCTATAATGCACGACAGAACAAAAATTAGACAGCTGCCTAACCTT
GAGGATACGAACATATCAGGTTATCAACCCATAACGTTATCAATGCAGAAAATGCACCAGGAGGACCCTACAAAATTGGAACCTCAGGGTCTTGCCCTA
TACCAATGGAAACGGATTTTTCGCAACAATGGCTTGGGCCGTCCCAAAAAACGACAAAAACAAAACAGCAACAAATCCATTAACAATAGAAGTACCATA
TGTACAGAAGGAGAAGACCAAATTACCGTTTGGGGGTTCCACTCTGACAACGAGACCCAAATGGCAAAGCTCTATGGGGACTCAAAGCCCCAGAAGTTC
CATCTGCCAACGGAGTGACCACACATTACGTTTCACAGATTGGTGGCTTCCCAAATCAAACAGAAGACGGAGGACTACCACAAAGTGGTAGAATTGTTG
TTACATGGTGCAAAAATCTGGGAAAACAGGAACAATTACCTATCAAAGGGGTATTTTATTGCCTCAAAAGGTGTGGTGCGCAAGTGGCAGGAGCAAGGT
AAAGGATCCTTGCCTTTAATTGGAGAAGCAGATTGCCTCCACGAAAAATACGGTGGATTAAACAAAAGCAAGCCTTACTACACAGGGGAACATGCAAAG
TAGGAAATTGCCCAATATGGGTGAAAACACCCTTGAAGCTGGAA
AAAGCAGGGGATAATTCTATTAACCATGAAGACTATCATTGCTTTGAGCTACATTCTATGTCTGGTTTTCGCTCAAAAACTTCCTGGAAATGACAACAG
GCAACGCTGTGCCTTGGGCACCATGCAGTACCAAACGAACGATAGTGAAAACAATCACGAATGACCAAATTGAAGTTACTAATGCTACTGAGCTGGTTC
TTCCTCAACAGGTGAAATATGCGACAGTCCTCATCAGATCCTTGATGGAAAAAACTGCACACTAATAGATGCTCTATTGGGAGACCCTCAGTGTGATGG
CAAAATAAGAAATGGGACCTTTTTGTTGAACGCAGCAAAGCCTACAGCAACTGTTACCCTTATGATGTGCCGGATTATGCCTCCCTTAGGTCACTAGTT
CATCCGGCACACTGGAGTTTAACAATGAAAGCTTCAATTGGACTGGAGTCACTCAAAACGGAACAAGCTCTGCTTGCATAAGGAGATCTAAAAACAGTT
TAGTAGATTGAATTGGTTGACCCACTTAAACTTCAAATACCCAGCATTGAACGTGACTATGCCAAACAATGAACAATTTGACAAATTGTACATTTGGGG
CACCACCCGGGTACGGACAAAGACCAAATCTTCCTGTATGCTCAAGCATCAGGAAGAATCACAGTCTCTACCAAAAGAAGCCAACAAACCGTAAGCCCG
TCGGATCTAGACCCAGAGTAAGGAATATCCCTAGCAGAATAAGCATCTATTGGACAATAGTAAAACCGGGAGACATACTTTTGATTAACAGCACAGGGA
AATTGCTCCTAGGGGTTACTTCAAAATACGAAGTGGGAAAAGCTCAATAATGAGATCAGATGCACCCATTGGCAAATGCAATTCTGAATGCATCACTCC
GGAAGCATTCCCAATGACAAACCATTCCAAAATGTAAACAGGATCACATACGGGGCCTGTCCCAGATATGTTAAGCAAAACACTCTGAAATTGGCAACA
TGCGAAATGTACCAGAGAAGAAAACACCCTTGAAGCTGGAA
Enter names of Sequences:t c p b
{'t':
'AAAGCAGGGGATAATTCTATTAACCATGAAGACTATCATTGCTTTGAGCTACATTCTATGTCTGGTTTTCGCTCAAAAACTTCCTGGAAATGACAACA
GGCAACGCTGTGCCTTGGGCACCATGCAGTACCAAACGGAACGATAGTGAAAACAATCACGAATGACCAAATTGAAGTTACTAATGCTACTGAGCTGGT
AGTTCCTCAACAGGTGAAATATGCGACAGTCCTCATCAGATCCTTGATGGAAAAAACTGCACACTAATAGATGCTCTATTGGGAGACCCTCAGTGTGAT
TCCAAAATAAGAAATGGGACCTTTTTGTTGAACGCAGCAAAGCCTACAGCAACTGTTACCCTTATGATGTGCCGGATTATGCCTCCCTTAGGTCACTAG
CTCATCCGGCACACTGGAGTTTAACAATGAAAGCTTCAATTGGACTGGAGTCACTCAAAACGGAACAAGCTCTGCTTGCATAAGGAGATCTAAAAACAG
TTTAGTAGATTGAATTGGTTGACCCACTTAAACTTCAAATACCCAGCATTGAACGTGACTATGCCAAACAATGAACAATTTGACAAATTGTACATTTGG
TTCACCACCCGGGTACGGACAAAGACCAAATCTTCCTGTATGCTCAAGCATCAGGAAGAATCACAGTCTCTACCAAAAGAAGCCAACAAACCGTAAGCC
TATCGGATCTAGACCCAGAGTAAGGAATATCCCTAGCAGAATAAGCATCTATTGGACAATAGTAAAACCGGGAGACATACTTTTGATTAACAGCACAGG
CTAATTGCTCCTAGGGGTTACTTCAAAATACGAAGTGGGAAAAGCTCAATAATGAGATCAGATGCACCCATTGGCAAATGCAATTCTGAATGCATCACT
ATGGAAGCATTCCCAATGACAAACCATTCCAAAATGTAAACAGGATCACATACGGGGCCTGTCCCAGATATGTTAAGCAAAACACTCTGAAATTGGCAA
GATGCGAAATGTACCAGAGAAGAAAACACCCTTGAAGCTGGAA', 'c':
'ATGAAGGCAATACTAGTAGTTCTGCTATATACATTTGCAACCGCAAATGCAGACACATTATGTATAGGTTATCATGCGAACAATTCAACAGACACTGT
ACAGTACTAGAAAAGAATGTAACAGTAACACACTCTGTTAACCTTCTAGAAGACAAGCATAACGGGAAACTATGCAAACTAAGAGGGGTAGCCCCATTG
TGGGTAAATGTAACATTGCTGGCTGGATCCTGGGAAATCCAGAGTGTGAATCACTCTCCACAGCAAGCTCATGGTCCTACATTGTGGAAACACCTAGTT
CAATGGAACGTGTTACCCAGGAGATTTCATCGATTATGAGGAGCTAAGAGAGCAATTGAGCTCAGTGTCATCATTTGAAAGGTTTGAGATATTCCCCAA
AGTTCATGGCCCAATCATGACTCGAACAAAGGTGTAACGGCAGCATGTCCTCATGCTGGAGCAAAAAGCTTCTACAAAAATTTAATATGGCTAGTTAAA
GAAATTCATACCCAAAGCTCAGCAAATCCTACATTAATGATAAAGGGAAAGAAGTCCTCGTGCTATGGGGCATTCACCATCCATCTACTAGTGCTGACC
AAGTCTCTATCAGAATGCAGATGCATATGTTTTTGTGGGGTCATCAAGATACAGCAAGAAGTTCAAGCCGGAAATAGCAATAAGACCCAAAGTGAGGGR
GAAGGGAGAATGAACTATTACTGGACACTAGTAGAGCCGGGAGACAAAATAACATTCGAAGCAACTGGAAATCTAGTGGTACCGAGATATGCATTCGCA
AAAGAAATGCTGGATCTGGTATTATCATTTCAGATACACCAGTCCACGATTGCAATACAACTTGTCAAACACCCAAGGGTGCTATAAACACCAGCCTCC
TCAGAATATACATCCGATCACAATTGGAAAATGTCCAAAATATGTAAAAAGCACAAAATTGAGACTGGCCACAGGATTGAGGAATATCCCGTCTATTCA
AGAGGCCTATTTGGGGCCATTGCCGGTTTCATTGAAGGGGGGTGGACAGGGATGGTAGATGGATGGTACGGTTATCACCATCAAAATGAGCAGGGGTCA
ATGCAGCCGACCTGAAGAGCACACAGAATGCCATTGACGAGATTACTAACAAAGTAAATTCTGTTATTGAAAAGATGAATACACAGTTCACAGCAGTAG
'p':
'AGCAGAAGCAGAGCATTTTCTAATATCCACAAAATGAAGGCAATAATTGTACTACTCATGGTAGTAACATCCAATGCAGATCGAATCTGCACTGGGAT
TCGTCAAACTCACCACATGTCGTCAAAACTGCTACTCAAGGGGAGGTCAATGTGACTGGTGTAATACCACTGACAACAACACCCACCAAATCTCATTTT
ATCTCAAAGGAACAGAAACCAGGGGGAAACTATGCCCAAAATGCCTCAACTGCACAGATCTGGACGTAGCCTTGGGCAGACCAAAATGCACGGGGAAAA
CTCGGCAAGAGTTTCAATACTCCATGAAGTCAGACCTGTTACATCTGGGTGCTTTCCTATAATGCACGACAGAACAAAAATTAGACAGCTGCCTAACCT
CGAGGATACGAACATATCAGGTTATCAACCCATAACGTTATCAATGCAGAAAATGCACCAGGAGGACCCTACAAAATTGGAACCTCAGGGTCTTGCCCT
TTACCAATGGAAACGGATTTTTCGCAACAATGGCTTGGGCCGTCCCAAAAAACGACAAAAACAAAACAGCAACAAATCCATTAACAATAGAAGTACCAT
TTGTACAGAAGGAGAAGACCAAATTACCGTTTGGGGGTTCCACTCTGACAACGAGACCCAAATGGCAAAGCTCTATGGGGACTCAAAGCCCCAGAAGTT
TCATCTGCCAACGGAGTGACCACACATTACGTTTCACAGATTGGTGGCTTCCCAAATCAAACAGAAGACGGAGGACTACCACAAAGTGGTAGAATTGTT
ATTACATGGTGCAAAAATCTGGGAAAACAGGAACAATTACCTATCAAAGGGGTATTTTATTGCCTCAAAAGGTGTGGTGCGCAAGTGGCAGGAGCAAGG
AAAAGGATCCTTGCCTTTAATTGGAGAAGCAGATTGCCTCCACGAAAAATACGGTGGATTAAACAAAAGCAAGCCTTACTACACAGGGGAACATGCAAA
ATAGGAAATTGCCCAATATGGGTGAAAACACCCTTGAAGCTGGAA', 'b':
'AAAGCAGGGGATAATTCTATTAACCATGAAGACTATCATTGCTTTGAGCTACATTCTATGTCTGGTTTTCGCTCAAAAACTTCCTGGAAATGACAACA
GGCAACGCTGTGCCTTGGGCACCATGCAGTACCAAACGAACGATAGTGAAAACAATCACGAATGACCAAATTGAAGTTACTAATGCTACTGAGCTGGTT
GTTCCTCAACAGGTGAAATATGCGACAGTCCTCATCAGATCCTTGATGGAAAAAACTGCACACTAATAGATGCTCTATTGGGAGACCCTCAGTGTGATG
CCAAAATAAGAAATGGGACCTTTTTGTTGAACGCAGCAAAGCCTACAGCAACTGTTACCCTTATGATGTGCCGGATTATGCCTCCCTTAGGTCACTAGT
TCATCCGGCACACTGGAGTTTAACAATGAAAGCTTCAATTGGACTGGAGTCACTCAAAACGGAACAAGCTCTGCTTGCATAAGGAGATCTAAAAACAGT
TTAGTAGATTGAATTGGTTGACCCACTTAAACTTCAAATACCCAGCATTGAACGTGACTATGCCAAACAATGAACAATTTGACAAATTGTACATTTGGG
TCACCACCCGGGTACGGACAAAGACCAAATCTTCCTGTATGCTCAAGCATCAGGAAGAATCACAGTCTCTACCAAAAGAAGCCAACAAACCGTAAGCCC
ATCGGATCTAGACCCAGAGTAAGGAATATCCCTAGCAGAATAAGCATCTATTGGACAATAGTAAAACCGGGAGACATACTTTTGATTAACAGCACAGGG
TAATTGCTCCTAGGGGTTACTTCAAAATACGAAGTGGGAAAAGCTCAATAATGAGATCAGATGCACCCATTGGCAAATGCAATTCTGAATGCATCACTC
TGGAAGCATTCCCAATGACAAACCATTCCAAAATGTAAACAGGATCACATACGGGGCCTGTCCCAGATATGTTAAGCAAAACACTCTGAAATTGGCAAC
ATGCGAAATGTACCAGAGAAGAAAACACCCTTGAAGCTGGAA'}
Percentages of A, C, G and T for t are : 34.14179104477612 22.108208955223883 20.42910447761194 23
.32089552238806
Total number of CGs in sequence:  21
Percentages of A, C, G and T for c are : 34.76848090982941 19.740048740861088 22.095857026807472 2
3.23314378554021
Total number of CGs in sequence:  20
Percentages of A, C, G and T for p are : 35.47486033519553 22.905027932960895 20.949720670391063 2
0.670391061452513
Total number of CGs in sequence:  22
Percentages of A, C, G and T for b are : 34.173669467787114 22.128851540616246 20.354808590102706
23.34267040149393
Total number of CGs in sequence:  21
```

We can see that the sample that matches our test most is B, Brisbane. While the percentages are not exact this is to be expected as bacteria can have mutations which can make an exact match hard to find. We can confirm this by looking at the total number of CG which we see is 21 in both sequences!