# Jacek Radajewski
# Student number: 43612772
# SCIE2100 Prac 4

## Questions 1 & 2

### *Q1*

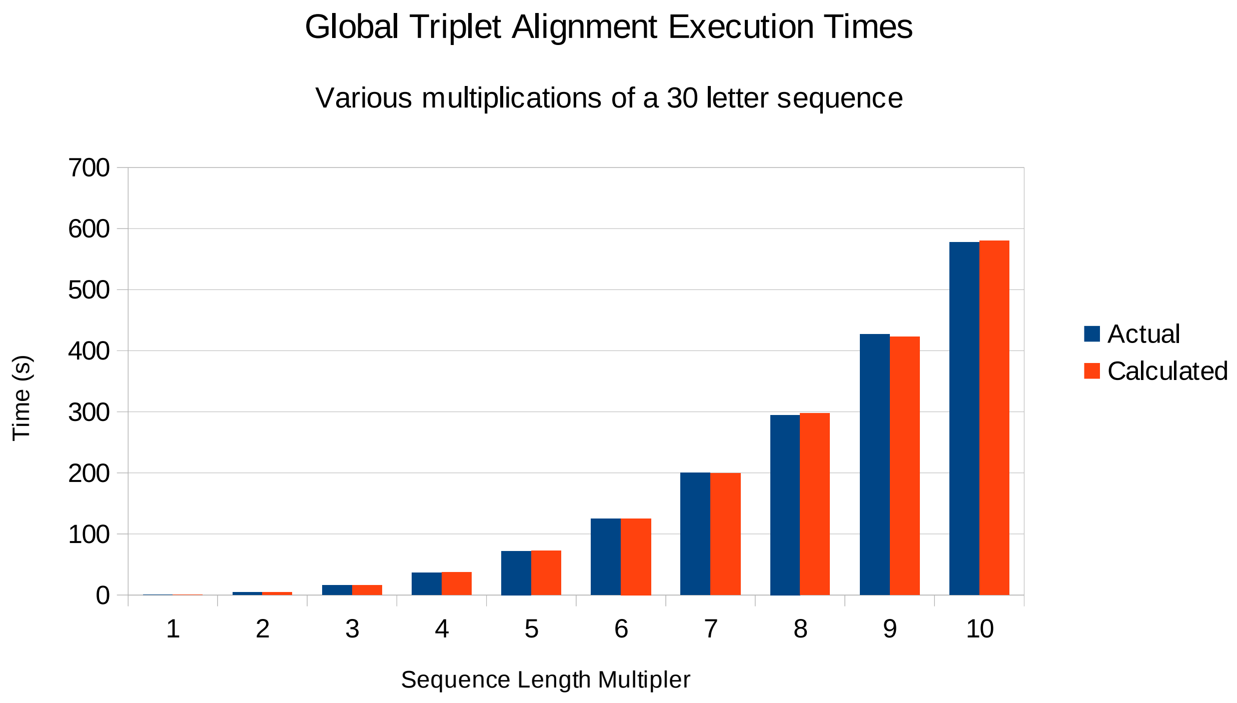
Time taken to run the triplet alignment was approximately 0.62 seconds. Pairwise alignment on the other hand completed in 0.005 seconds. Please refer to the code and execution output sections.

### *Q2*

In this question we have doubled the sequence length and it has taken 4.74 seconds to calculate the global alignment of the triplet. The time taken to execute the algorithm has increased by 4.74 / 0.62 which is around 7.65. In triplet alignment we are processing a three dimensional matrix so by increasing the the length of our sequence by a factor of 2 we increase the total number of the cells in the matrix by a factor of $2^3 = 8$. The actual increase in execution time was 7.65.

The performance of the algorithm is $O(n^3)$.

I have written code which calculates the execution time for computing triplet global alignment for sequences of length from n to 10n. These were graphed next to expected execution times and shown below. The expected calculations for 2 <= n <= 10 is based on n=1.

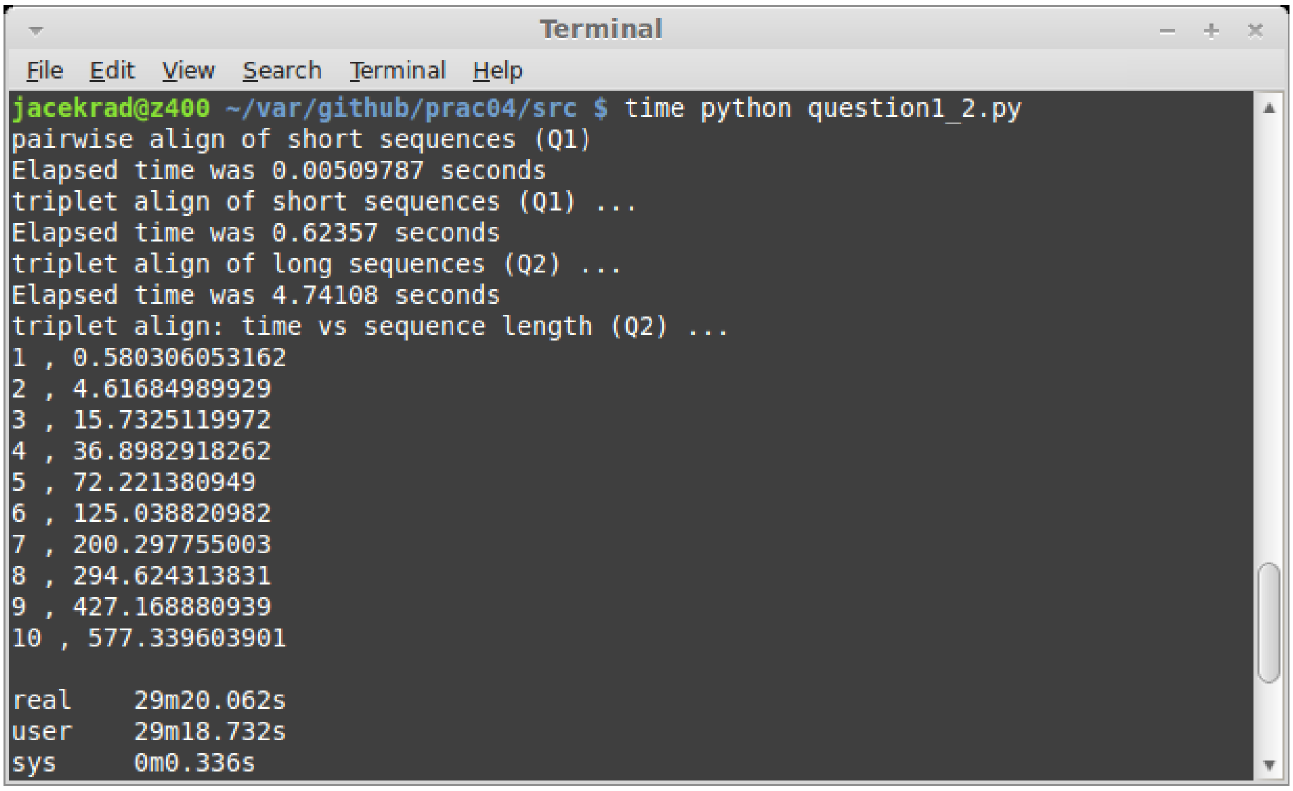


## *Code*

```
'''
Created on 15/04/2014

@author: s4361277
'''
from sequence import *
import time

seqA1 = Sequence("AGCGCGATTATATAAGACGGACGGCTAAAG")
seqB1 = Sequence("AGCGGATATTTATATCGCACGACGACTACG")
seqC1 = Sequence("GGATCGATTATATAGCCTGGACGAGACATG")


seqA2 = Sequence("AGCGCGATTATATAAGACGGACGAGACGACAGCGCGATAGACGAGACGGACGAGACGACT")
seqB2 = Sequence("GGATCGATTATATAGCCTGGACGAGACATGGGATCGAAGACGACCTGGACGAGACATGAC")
seqC2 = Sequence("AGCGGATATTTATATCGCACGACGACTACGAGCGGAAGACGAGTTTCGCACGACACTACG")

matrix = readSubstMatrix("dna.matrix", DNA_Alphabet)

print "pairwise align of short sequences (Q1)"
start_time = time.time()
alignGlobal(seqA1, seqB1, matrix, -1)
end_time = time.time()
print("Elapsed time was %g seconds" % (end_time - start_time))

print "triplet align of short sequences (Q1) ..."
start_time = time.time()
tripletAlignGlobal(seqA1, seqB1, seqC1, matrix, -1)
end_time = time.time()
print("Elapsed time was %g seconds" % (end_time - start_time))

print "triplet align of long sequences (Q2) ..."
start_time = time.time()
tripletAlignGlobal(seqA2, seqB2, seqC2, matrix, -1)
end_time = time.time()
print("Elapsed time was %g seconds" % (end_time - start_time))

# the following code provides a more detailed analysis of algorithm time
# the output was use to genmerate a comparison graph of expected vs real
# execution times

print "triplet align: time vs sequence length (Q2) ..."
sequence_string_fragment_1 = "AGCGCGATTATATAAGACGGACGGCTAAAG"
sequence_string_fragment_2 = "AGCGGATATTTATATCGCACGACGACTACG"
sequence_string_fragment_3 = "GGATCGATTATATAGCCTGGACGAGACATG"

sequence_string_1 = ""
sequence_string_2 = ""
sequence_string_3 = ""

for length_multiplier in range(1, 11):
    sequence_string_1 += sequence_string_fragment_1
    sequence_string_2 += sequence_string_fragment_2
    sequence_string_3 += sequence_string_fragment_3
    start_time = time.time()
    tripletAlignGlobal(Sequence(sequence_string_1), Sequence(sequence_string_2),
                        Sequence(sequence_string_3), matrix, -1)
    end_time = time.time()
    # output produced can be redirected into a CSV file
    print length_multiplier, ",", end_time - start_time
```

### Execution

Screen shot below shows the execution of the code for questions 1 and 2.



```
jacekrad@z400 ~/var/github/prac04/src $ time python question1_2.py
pairwise align of short sequences (Q1)
Elapsed time was 0.00509787 seconds
triplet align of short sequences (Q1) ...
Elapsed time was 0.62357 seconds
triplet align of long sequences (Q2) ...
Elapsed time was 4.74108 seconds
triplet align: time vs sequence length (Q2) ...
1 , 0.580306053162
2 , 4.61684989929
3 , 15.7325119972
4 , 36.8982918262
5 , 72.221380949
6 , 125.038820982
7 , 200.297755003
8 , 294.624313831
9 , 427.168880939
10 , 577.339603901

real	29m20.062s
user	29m18.732s
sys	0m0.336s
```

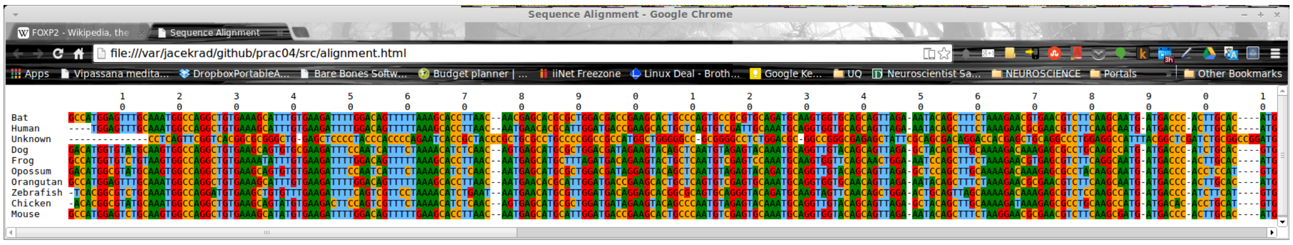
## Question 4

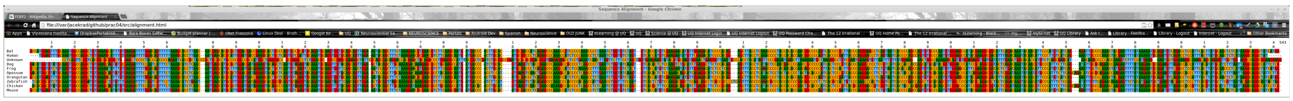
### a, b & c

All of the sequences in this alignment are from the highly conserved FOXP2 gene. One could conclude for the alignment to be reasonable if there are few gaps and the conserved regions are visible in the HTML output. Looking at the HTML representation of the alignment (see below) we can clearly see that only few gaps are present and from the colours matches we can conclude that the sequences are highly conserved. With the exception of the “unknown” sequence all sequences are highly conserved with many regions identical or almost identical.

### Alignment

The screen grabs are shown. First is of the head of the alignment and the second show full length of the alignment.



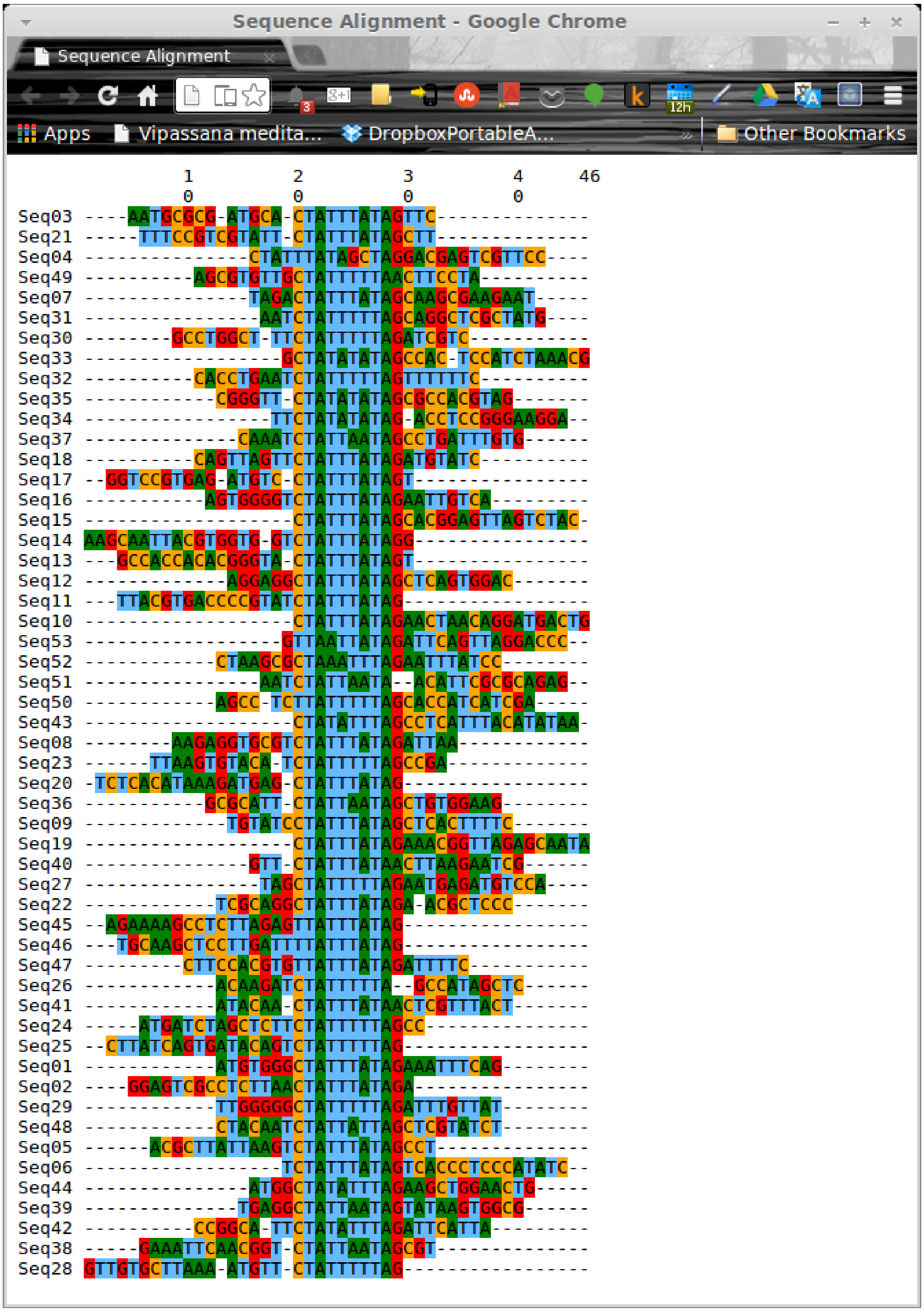


## *Code*

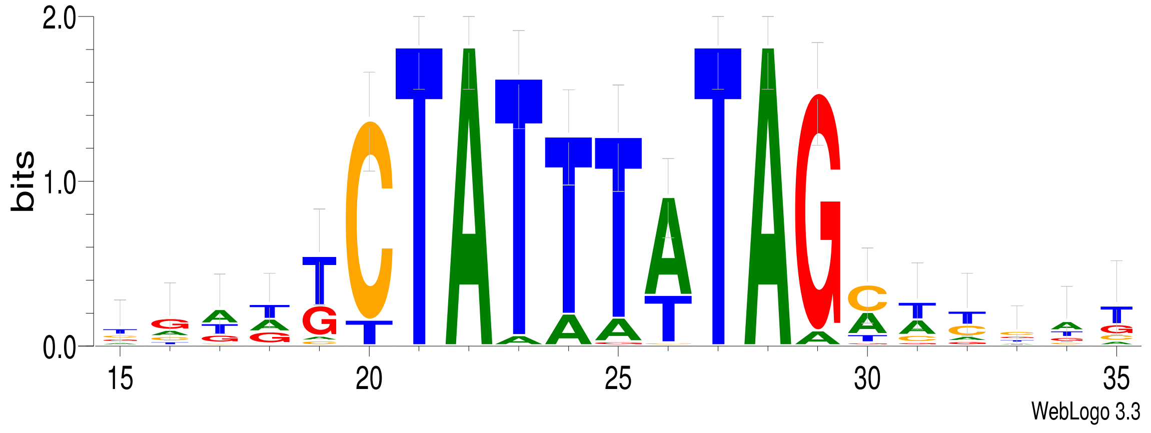
```
'''
Created on 15/04/2014

@author: s4361277
'''
from sequence import *
import time

multi_align_sequences = readFastaFile("multiAlign.fasta", DNA_Alphabet)

start_time = time.time()
alignment = runClustal(multi_align_sequences)
end_time = time.time()
print("Elapsed time was %g seconds" % (end_time - start_time))

alignment.writeClustal("multiAlign.aln")
alignment.writeHTML("alignment.html")
```

## Question 6

Two screen shots are provided.  First shows the HTML representation of the alignment clearly showing the similar region.  Second is the logo coloured similarly to the HTML file and  “zoomed” on the motif region.



```
               1         2         3         4    46
               0         0         0         0
Seq03 ----AATGCGCG-ATGCA-CTATTTATAGTTC--------------
Seq21 -----TTTCCGTCGTATT-CTATTTATAGCTT--------------
Seq04 ---------------CTATTTATAGCTAGGACGAGTCGTTCC----
Seq49 ----------AGCGTGTTGCTATTTTTAACTTCCTA----------
Seq07 ---------------TAGACTATTTATAGCAAGCGAAGAAT-----
Seq31 ----------------AATCTATTTTTAGCAGGCTCGCTATG----
Seq30 --------GCCTGGCT-TTCTATTTTTAGATCGTC-----------
Seq33 ------------------GCTATATATAGCCAC-TCCATCTAAACG
Seq32 ----------CACCTGAATCTATTTTTAGTTTTTTC----------
Seq35 ------------CGGGTT-CTATATATAGCGCCACGTAG-------
Seq34 -----------------TTCTATATATAG-ACCTCCGGGAAGGA--
Seq37 --------------CAAATCTATTAATAGCCTGATTTGTG------
Seq18 ----------CAGTTAGTTCTATTTATAGATGTATC----------
Seq17 --GGTCCGTGAG-ATGTC-CTATTTATAGT----------------
Seq16 -----------AGTGGGGTCTATTTATAGAATTGTCA---------
Seq15 -------------------CTATTTATAGCACGGAGTTAGTCTAC-
Seq14 AAGCAATTACGTGGTG-GTCTATTTATAGG----------------
Seq13 ---GCCACCACACGGGTA-CTATTTATAGT----------------
Seq12 -------------AGGAGGCTATTTATAGCTCAGTGGAC-------
Seq11 ---TTACGTGACCCCGTATCTATTTATAG-----------------
Seq10 -------------------CTATTTATAGAACTAACAGGATGACTG
Seq53 ------------------GTTAATTATAGATTCAGTTAGGACCC--
Seq52 ------------CTAAGCGCTAAATTTAGAATTTATCC--------
Seq51 ----------------AATCTATTAATA--ACATTCGCGCAGAG--
Seq50 ------------AGCC-TCTTATTTTTAGCACCATCATCGA-----
Seq43 -------------------CTATATTTAGCCTCATTTACATATAA-
Seq08 --------AAGAGGTGCGTCTATTTATAGATTAA------------
Seq23 ------TTAAGTGTACA-TCTATTTTTAGCCGA-------------
Seq20 -TCTCACATAAAGATGAG-CTATTTATAG-----------------
Seq36 -----------GCGCATT-CTATTAATAGCTGTGGAAG--------
Seq09 -------------TGTATCCTATTTATAGCTCACTTTTC-------
Seq19 -------------------CTATTTATAGAAACGGTTAGAGCAATA
Seq40 ---------------GTT-CTATTTATAACTTAAGAATCG------
Seq27 ----------------TAGCTATTTTTAGAATGAGATGTCCA----
Seq22 ------------TCGCAGGCTATTTATAGA-ACGCTCCC-------
Seq45 --AGAAAAGCCTCTTAGAGTTATTTATAG-----------------
Seq46 ---TGCAAGCTCCTTGATTTTATTTATAG-----------------
Seq47 ---------CTTCCACGTGTTATTTATAGATTTTC-----------
Seq26 ------------ACAAGATCTATTTTTA--GCCATAGCTC------
Seq41 ------------ATACAA-CTATTTATAACTCGTTTACT-------
Seq24 -----ATGATCTAGCTCTTCTATTTTTAGCC---------------
Seq25 --CTTATCAGTGATACAGTCTATTTTTAG-----------------
Seq01 ------------ATGTGGGCTATTTATAGAAATTTCAG--------
Seq02 ----GGAGTCGCCTCTTAACTATTTATAGA----------------
Seq29 ------------TTGGGGGCTATTTTTAGATTTGTTAT--------
Seq48 ------------CTACAATCTATTATTAGCTCGTATCT--------
Seq05 ------ACGCTTATTAAGTCTATTTATAGCCT--------------
Seq06 ------------------TCTATTTATAGTCACCCTCCCATATC--
Seq44 ---------------ATGGCTATATTTAGAAGCTGGAACTG-----
Seq39 --------------TGAGGCTATTAATAGTATAAGTGGCG------
Seq42 ----------CCGGCA-TTCTATATTTAGATTCATTA---------
Seq38 -----GAAATTCAACGGT-CTATTAATAGCGT--------------
Seq28 GTTGTGCTTAAA-ATGTT-CTATTTTTAG-----------------
```



## Question 7

Consensus sequence for the highly conserved region is CTATTTATAG.
I have used two regular expressions The first (**"CTAT[AT]{3}TAG"**) was a strict version and the second (**"[CT]TA[TA]{4}TA[GA\-]"**) is much more flexible. Both regular expressions will match a sequence of 10 letters which corresponds to the highly conserved region (see logo position 20-29).

Table below provides short justification for each position/column in both regular expressions.

| Position | Strict | Flexible |
|---|---|---|
| 20 | Mainly C, excluding Ts | Mainly Cs, but some Ts present |
| 21 | Only T | Only T |
| 22 | Only A | Only A |
| 23 | Almost all Ts | Few, but some As present. Included with Ts |
| 24 | Mainly Ts, but also As | Mainly Ts, but also As |
| 25 | Mainly Ts, but also As | Mainly Ts, but also As |
| 26 | Mostly A, but a lot of Ts too | Mostly A, but a lot of Ts too |
| 27 | T's only | T's only |
| 28 | As only | As only |
| 29 | Almost all Gs, excluding rest | Mostly Gs but 2 As and 2 gaps present |

See question 10 code for how the above were applied.

## Question 9

Below is the position weight matrix generated (see question 10 code) from position 20 to position 29 based on highly conserved region visible (identified by eye) in the logo generated in question 7.

```
A   -25.33 -25.33  +1.39  -1.89  -0.50  -0.79  +0.97 -25.33  +1.39
C    +1.27 -25.33 -25.33 -25.33 -25.33 -25.33  -2.58 -25.33 -25.33
G   -25.33 -25.33 -25.33 -25.33 -25.33  -2.58 -25.33 -25.33 -25.33
T    -0.79  +1.39 -25.33  +1.35  +1.22  +1.24  +0.25  +1.39 -25.33
```

## Question 10

Search code and results for both regular expression as well as the PWM are show below.

| Method | Number of matches |
|---|---|
| Strict regular expression | 2 |
| Flexible regular expression | 10 |
| Position Weight Matrix | 14 |

It is clear that the PWM has matched many more sequences than either of the regular expressions which was expected as it was generated by mathematical means rather than visual inspection.

Unlike regular expressions, PWM also returns partial matches with a match score as the last element of the result list.  This feature of PWM allows for more  flexible representation of results and perhaps more suitable to degenerate motifs where the level of similarity may vary from sequence to sequence.

Regular expressions may have benefits over PWM during initial analysis as they allow for manual manipulation of the search string (regular expression).  One can manually try various different regular expressions to see which is best suited and returns best search results.

### *Code*

```
'''
Created on 15/04/2014

@author: s4361277
'''
from sequence import *

alignment = readClustalFile("myAlign.aln", DNA_Alphabet)
pwm = PWM(alignment, start=19, end=28)
print pwm

# strict regular expression is closer to the consensus sequences
# but does allow for some flexibility
strict_regexp = Regexp("CTAT[AT]{3}TAG")
```

```
# flexible regular expression is more flexible than the strict regexp
# and allows for more alternative letter.  This regular expression should
# produce more matches.
flexi_regexp = Regexp("[CT]TA[TA]{4}TA[GA\-]")

sequences = readFastaFile("motifSearch.fasta", DNA_Alphabet)

for sequence in sequences:
    # result = flexi_regexp.search(sequence)
    pwm_results = pwm.search(sequence)
    strict_regexp_results = strict_regexp.search(sequence)
    flexi_regexp_results = flexi_regexp.search(sequence)
    print
    print "-------------------------", sequence.name,
"------------------------------"
    print "REGEXP Strict :", strict_regexp_results
    print "REGEXP Flexi  :", flexi_regexp_results
    print "PWM           :", pwm_results
```

### *Search Results*

```
------------------------- extracted07 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : []
PWM           : [(134, 'GCTATATTTAT', 6.1057151718246043)]

------------------------- extracted08 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : []
PWM           : [(115, 'GCTATATTTAT', 6.1057151718246043)]

------------------------- extracted09 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : []
PWM           : []

------------------------- J04111 ------------------------------
REGEXP Strict : [(139, 'CTATTTTTAG', 1.0)]
REGEXP Flexi  : [(139, 'CTATTTTTAG', 1.0)]
PWM           : [(138, 'GCTATTTTTAG', 12.407647098418472)]

------------------------- L36125 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : [(33, 'CTAAAAATAG', 1.0)]
PWM           : [(32, 'GCTAAAAATAG', 6.1270004883357654)]

------------------------- M29660 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : [(63, 'CTAAAAATAG', 1.0)]
PWM           : [(62, 'ACTAAAAATAG', 5.0578020848739484)]

------------------------- M61126 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : [(10, 'CTAAAAATAG', 1.0)]
PWM           : [(9, 'GCTAAAAATAG', 6.1270004883357654)]

------------------------- M62404 ------------------------------
```

```
REGEXP Strict : []
REGEXP Flexi  : [(173, 'TTAAAAATAA', 1.0)]
PWM           : [(172, 'ATTAAAAATAA', 0.37061312294387538)]

------------------------- M84685 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : [(174, 'CTATATATAA', 1.0)]
PWM           : [(173, 'ACTATATATAA', 7.7045616348512986)]

------------------------- U18131 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : [(83, 'CTATATATAA', 1.0)]
PWM           : [(82, 'ACTATATATAA', 7.7045616348512986)]

------------------------- X00371 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : []
PWM           : []

------------------------- X04260 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : [(54, 'CTAAATATAG', 1.0)]
PWM           : [(53, 'ACTAAATATAG', 7.094684012134989)]

------------------------- X12447 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : [(77, 'CTAAATATAG', 1.0)]
PWM           : [(76, 'CCTAAATATAG', 7.094684012134989)]

------------------------- X57155 ------------------------------
REGEXP Strict : [(134, 'CTATTTTTAG', 1.0)]
REGEXP Flexi  : [(134, 'CTATTTTTAG', 1.0)]
PWM           : [(133, 'GCTATTTTTAG', 12.407647098418472)]

------------------------- X71910 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : []
PWM           : [(130, 'GCTATATTTAT', 6.1057151718246043)]

------------------------- X85744 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : []
PWM           : [(132, 'TCTATATTTAT', 6.4961424025682284)]

------------------------- Z20656 ------------------------------
REGEXP Strict : []
REGEXP Flexi  : []
PWM           : []
```